3530 D & F

ORION *delta* DATA LOGGER

OPERATING MANUAL

Solartron Instruments

MP MASTER

MA20216

SOLARTRON

Schlumberger

Digitized by the Internet Archive
in 2025

https://archive.org/details/manualsplus_07239

3530 D & F
ORION *delta* DATA LOGGER
OPERATING MANUAL

Issue 3: March 1985.

Part No: 35302040.

SOLARTRON
Schlumberger

Solartron Instruments, Victoria Road, Farnborough
Hampshire, England GU14 7PW Telephone: Farnborough (0252) 544433
Telex: 858245 Solfar G Cables: Solartron Farnborough
A division of Schlumberger Electronics (UK) Ltd

Solartron pursues a policy of continuous development and product improvement
The specification in this document may therefore be changed without notice

19 © 83

Contents

Chapter 1 General Information *Page No.*

1	Introduction	1.2
2	Main Features	1.3
3	Inputs	1.4
3.1	Input Types	1.4
3.2	Analogue Inputs	1.4
3.3	Digital Inputs	1.5
4	Outputs	1.6
4.1	Digital Outputs	1.6
4.2	Analogue Outputs	1.7
5	Setting-up and Control	1.7
6	Power Supply and Power Failure	1.8
7	Front Panel Layout	1.9
8	Optional Input/Output Modules	1.10
9	Locations of Standard and Optional Modules	1.12
10	Peripheral Devices	1.13

Chapter 2 Using ORION

1	General Instructions for Using ORION	2.2
2	Example Setting-up Procedure	2.5
3	Safety	2.9

Chapter 3 Installation

1	Connecting the AC mains or DC supply	3.2
1.1	Safety Precautions	3.2
1.2	General	3.2
1.3	Procedure	3.2
2	Connecting Peripheral Devices	3.4
2.1	General	3.4
2.2	Controller pcb	3.5
2.3	GP-IB Interface (35304A)	3.7
2.4	Serial Interface (35304B)	3.10
3	Rack Mounting	3.13
4	Slave Expansion Unit 3530E	3.17

Chapter 4 Using the Reed Relay or Solid State Selectors (35301A/B) *Page No.*

1	General Information	4.2
1.1	Current and Resistance Measurement	4.2
1.2	Energising Currents	4.2
1.3	Strain Measurement	4.3
1.4	Common Mode and Series Mode Interference	4.4
1.5	Sense Amplifiers	4.7
1.6	Permitted inputs and sensors	4.7
1.7	Input Connectors	4.8
1.8	Channel Numbering	4.9
1.9	Selector Modules	4.12
1.10	Condensed Specification for Reed Relay and Solid State Selectors	4.12
2	Input Connections	4.13
2.1	General Procedure	4.13
2.2	Voltage Measurement	4.16
2.3	Thermocouple Measurement	4.18
2.4	2-terminal Resistance Measurement	4.22
2.5	4-terminal Resistance and PRT Measurement	4.24
2.6	Full Bridge Strain Gauge Measurement	4.26
2.7	¼-Bridge Strain Gauge Measurement	4.28
2.9	3-wire Strain Gauge Measurement	4.34
2.10	3-pole Current Measurement	4.38

Chapter 5 Using the Event-Status Input Module (35302A)

1	General Information	5.2
1.1	Introduction	5.2
1.2	Status Inputs	5.3
1.3	Event Inputs	5.4
1.4	Counting and Totalising Events	5.5
1.5	Setting up Alarms for Status and Event Inputs	5.5
1.6	Channel Numbering and Pin Assignments for Inputs	5.7
1.7	Input Specification	5.10
1.8	Input Energisation	5.11
2	Input Connections	5.12
2.1	General Procedure	5.12
2.2	Fitting additional capacitors for greater contact-bounce filtering	5.17

Chapter 6 Using the Digital Input-Output Module (35302B) *Page No.*

1	General Information	6.2
1.1	Introduction	6.2
1.2	Status Inputs	6.2
1.3	Alarm Outputs	6.3
1.4	Stimulus Outputs	6.3
1.5	External Measurement Modules	6.3
1.6	Setting up Alarms for Status Inputs	6.8
1.7	Channel Numbering and Pin Assignments	6.10
1.8	Avoiding multiple use of an Input or Output	6.11
2	Connecting the Inputs	6.12
2.1	General	6.12
2.2	Procedure	6.12
2.3	Specification	6.12
3	Connecting the Outputs	6.13
3.1	General	6.13
3.2	Procedure	6.14
3.3	Specification	6.16

Chapter 7 Using the Counter-Timer Module (35302C)

1	Introduction	7.1
2	The Measurement Functions	7.2
2.1	Frequency	7.2
2.2	Period	7.5
2.3	Totalise	7.7
2.4	Interval	7.8
3	Obtaining Measurements	7.9
4	Using the F and G Inputs	7.10
5	Input Connections and Switch Settings	7.11
6	Transducer Energisation	7.17
7	General Procedure for Making Connections and Switch Selection	7.18

Chapter 8 Using the Analogue Output Module (35301E)

1	Introduction	8.1
2	Setting up and Using the Outputs	8.2
3	Current Outputs: Connections and Switch Settings	8.4
4	Voltage Outputs: Connections and Switch Settings	8.5
5	Current/Voltage Selector Switches	8.6
6	PL1/PL2 Connections	8.7
7	Procedure	8.8

Chapter 9 Front Panel Controls *Page No.*

1	Introduction	9.2
2	Location of Controls	9.2
3	Function Keys	9.3
3.1	Control Keys	9.4
	RUN	9.4
	PAUSE	9.4
	MONitor	9.4
	LOCAL	9.5
3.2	Definition Keys	9.5
	CHANnel	9.5
	TASK	9.11
	CONVersion	9.14
	TEXT	9.14
	INITialise	9.15
	TAPE	9.15
	LIST	9.17
	PAPER ADVance	9.17
3.3	Edit Keys	9.17
	↑ ↓	9.17
	← →	9.18
	CHARacter INSERT	9.18
	CHARacter DELETE	9.18
	LINE INSERT	9.18
	LINE DELETE	9.18
3.4	Processing Keys – see Chapter 12	
4	Keypad	9.19
5	Soft Keys	9.20
6	Keyswitch	9.21
7	Display	9.21
8	Scrolling Facility	9.22
9	Power Annunciators	9.22
10	Audible Indications	9.23

Chapter 10 Data Reduction Facilities

1	Introduction	10.2
2	Conversions	10.4
3	Alarms	10.6
3.1	Examples	10.8

Chapter 11 Recording Devices and Output Formats

1	Introduction	11.2
2	Printer	11.3
2.1	General Information	11.3
2.2	Print-Out Formatting	11.4
3	Cartridge Recorder	11.7
4	Output Formats	11.11
4.1	Adding Text	11.11
4.2	General Details	11.12
4.3	Coded Format	11.17

Chapter 12 Processing

		Page No.
1	Processing Facilities	12.3
1.1	Introduction	12.3
1.2	Summary of Processing Facilities	12.3
2	Introduction to Processing Programs	12.4
2.1	Processing in the Task Sequence	12.6
2.2	Line Numbers	12.8
2.3	Definition of a Variable	12.9
2.4	Introduction to ORION Variables	12.9
2.5	Creating and Running a Simple Program	12.10
2.6	Obtaining a Listing of the Program	12.12
3	Entering and Editing Programs	12.13
3.1	Entering Programs	12.13
3.2	Single and Multi-statement Lines	12.13
3.3	Statement Separators/Delimiters	12.14
3.4	Expressions	12.14
3.5	Assignments	12.14
3.6	The Processing Keys	12.15
3.7	The Edit Keys	12.15
3.8	Resequencing	12.18
4	Syntax and Error Information	12.19
5	Constants	12.20
6	Types of Variable	12.21
6.1	Simple Variables	12.22
6.2	Indexed Variables	12.22
6.3	Pre-defined Variables	12.24
6.4	User-created Variables	12.26
6.5	Global Variables	12.27
6.6	Rules Governing Variable Names	12.27
6.7	Applications of Pre-defined and User-created Variables	12.28
7	Mathematical Operators and Functions	12.32
7.1	Mathematical Operators and Priorities	12.32
7.2	Trigonometric and Exponential Functions	12.33
7.3	Relational and Boolean Operators	12.34
8	Control Structures	12.34
8.1	IF . . . THEN Statements	12.34
8.2	FOR . . . NEXT Loops	12.36
8.3	GOTO	12.38
9	Output Statements	12.39
9.1	Introduction	12.39
9.2	Output Formatting	12.39
9.3	Print Statements	12.40
9.4	Print Statement Extensions	12.45
9.5	Output Format Declaration	12.48
9.6	Result Numbering	12.50
9.7	Comparison of Print Statement Extensions and Output Format Declaration	12.51
9.8	Write Statements	12.54
9.9	KILL	12.57
9.10	SET . . . TO	12.58
10	Digital Inputs	12.61
11	Control of Other Tasks	12.62
11.1	Triggering Other Tasks	12.62
11.2	Aborting Other Tasks	12.62

Chapter 12 Processing (continued) *Page No.*

12	Counters and Timers	12.63
12.1	Counters	12.63
12.2	Timers	12.64
13	Random Channel Access	12.64
14	Storing and Replaying Processed Data and Programs	12.68
15	Memory Capacity	12.69
16	Vocabulary of Functions	12.69
17	Example Programs	12.70
17.1	Contents	12.70
17.2	Introduction	12.71
17.3	Single Channel Average over 5 Readings	12.72
17.4	Average of 3 Channels on Every Scan, with a Cumulative Average of Each Channel Every 5 Scans	12.73
17.5	Maximum and Minimum Readings, on One Channel over 10 Scans	12.74
17.6	Output on Change, for One Channel	12.75
17.7	Output on Change, Giving Old and New Values and the Difference Between Them	12.76
17.8	Warning if Channel Reading is Outside Limits	12.77
17.9	Alarm Threshold	12.78
17.10	Walking Window Averages	12.79
17.11	Walking Window Average on One Channel	12.80
17.12	Walking Window Average on 3 Channels, over 5 Readings	12.81
17.13	Mean Reading and Standard Deviation of 20 Channels, Over 10 Scans	12.82

Chapter 13 Specification 13.1

Appendix 1 Error Messages A1.1

Chapter 1
General Information

Contents

		Page No.
1	Introduction	1.2
2	Main Features	1.3
3	Inputs	1.4
3.1	Input Types	1.4
3.2	Analogue Inputs	1.4
3.3	Digital Inputs	1.5
4	Outputs	1.6
4.1	Digital Outputs	1.6
4.2	Analogue Outputs	1.7
5	Setting-up and Control	1.7
6	Power Supply and Power Failure	1.8
7	Front Panel Layout	1.9
8	Optional Input/Output Modules	1.10
9	Locations of Standard and Optional Modules	1.12
10	Peripheral Devices	1.13

1 INTRODUCTION

3530 ORION is a compact and versatile Data Logging System; it is contained in a single, easily-transported unit which may be bench- or rack-mounted. ORION contains powerful software which provides a comprehensive and flexible logging capability.

The logger is offered in 2 versions: ORION *delta* (3530D) and ORION F (3530F). Both have essentially identical measurement and processing capability, and both may be controlled remotely.

ORION *delta* has a display with front panel controls, a built-in printer and DC100 cartridge recorder. Control and programming can be achieved using the front panel keys in conjunction with 'prompt' messages which appear on the integral display. In this way complex logging routines can be set-up without any previous programming knowledge. The built in cartridge recorder is used for storage of measurement data and setting-up routines, and the printer will give instant print-outs of measurement data and program listing. Control and programming can also be achieved from remote devices connected to one of the interfaces at the rear of the logger.

ORION F has a 'blank' front panel and is intended for remote-control only. A local vdu, connected to the Monitor Interface socket on the front panel, or a suitable device connected to one of the rear interfaces, may be used.

This manual includes details for using the ORION *delta* via its front panel. Note that all the facilities are also accessible by means of remote control. To control ORION remotely, refer also to the accompanying handbook 'Communicating with ORION *delta*'; this contains details on the interfaces and the command language.

2 MAIN FEATURES

ORION may be powered from ac mains or an external battery. The System will accept a wide range of analogue and digital inputs which can be intermixed. Digital outputs are available to give alarm information or to provide stimuli to control experiments or test rigs. Analogue outputs are available for controlling such devices as plotters and pen recorders.

All setting-up and operation is achieved using the front panel which contains control keys for the main functions. When a function is selected, the display provides a sequence of 'menu's' which are associated with each step in the setting-up. A selection from the menu is made by pressing the related 'soft key' below the display. Each soft key therefore assumes a different meaning, depending on the displayed options. Maximum flexibility in setting-up is thereby provided with only a minimum of front panel controls. The automatic menu selection permits fast and easy setting-up, without need for programming or software knowledge.

The built-in magnetic cartridge recorder can be used to record and replay logged data. It may also be used to store and retrieve setting-up routines. A built-in paper-strip printer is fitted which will give an instant record of logged data or setting-up conditions.

Powerful processing and data reduction facilities are provided. Some, such as the conversion of measurements to temperature or strain, occur automatically. Others, such as the application of non-standard units, and linear and polynomial equations may be defined and entered by the user.

In the case of strain gauge measurement, initial resistance and bridge imbalance is measured and stored (as part of the initialising routine) for subsequent use in the calculation of strain. Alarm conditions can be created by setting up limits and thresholds.

Alarm states can be used for controlling functions in addition to data recording.

The setting-up procedure for a logging program can be summarised as follows:

1. The details for all channels are entered as 'Channel Definitions'.
2. Groups of channels are allocated to logging tasks in 'Task Definitions'. Up to 8 tasks can be set-up, each with its own processing program.

The tasks can be separate logging routines, or they can be made interdependent by suitable cross triggering; alarm channels and digital 'event' channels can be used to effect triggering of a specific task. Tasks can be used for both logging and for control of digital outputs.

The setting-up facilities in ORION thus permit very flexible and comprehensive logging routines to be easily entered. Furthermore, the user may readily change, or refer back to, the settings already in the logger.

ORION includes a monitoring facility which permits the user to obtain an instantaneous display of current readings from any chosen channels, or task outputs. This facility operates at 'low priority' and does not significantly affect logging tasks that may be running.

A wide range of peripheral devices may be connected to ORION using the standard and optional interfaces offered. Such devices include a VDU or terminal, external printers, and bulk storage devices.

An RS232 interface is provided as standard on all loggers. One of two optional plug-in interfaces may be fitted; the GP-IB Interface (35304A) permits any GP-IB-compatible device to be connected, the Serial Interface (35304B) permits any RS422-(or RS232-) compatible device to be connected.

3 INPUTS

3.1 INPUT TYPES

The following types of input can be connected.

Analogue-type inputs
Voltage and current, ac and dc
Resistance
Thermocouples and PRT's
Strain Gauges in full-, ½- or ¼-bridge configurations

Digital-type inputs
Status of external devices such as plant switches
Detection of events as they occur
Timing of events
Event counting
Inputs grouped in bcd or binary bytes
Frequency, Totalising, Period and Interval measurement

3.2 ANALOGUE INPUTS

Analogue measurement is performed by the analogue-to-digital converter (adc) which employs the pulse-width conversion technique. This is a well-proven adc technique which offers fast and accurate measurement. Each measurement by the adc is a continuous integration, with a duration of 150μs (minimum). The overall time for the logging system to take measurements is necessarily longer owing to settling times and the requirement for sufficient integrations of each measurement to be performed.

Integration times and reading rates Four different integration times are offered giving a choice of reading rates and measurement resolution. Generally the shorter integration times give faster measurements but with reduced scale length. The longer integration times give slower measurements but with increased sensitivity and better interference rejection. The shortest integration time, for instance, gives 500 readings/s with the Solid State Selector, or 150 readings/s with the Reed Selector. The longer integration times of 20ms and 80ms give reading rates of 40/s and 10/s respectively, both of which provide good interference rejection. The integration times are set automatically according to the 'reading rate' requested (part of Task Definition).

These reading rates are applicable to DC voltage-based measurements (e.g. thermocouples), and are the effective rates at which the measurements are accepted by the memory.

A 1000-measurement buffer is used for temporary storage of results; when this is filled, the average scan rate will be slower to accommodate any process or output device throughput.

Auto-Zero An auto-zero and calibration sequence ensures that the highest possible accuracy of measurement is attained; this makes periodic measurements of zero and reference values. Zeroes are measured for each range at appropriate resolutions, and stored. These readings are used as corrections when the active channels are scanned. The auto-zero and calibration measurements are performed automatically and internally, no input channels are used for this purpose.

Thermal Offset Thermal-offset correction is offered on resistance-type measurements. This is useful in cases where unavoidable thermal emf's are produced, e.g. in wiring, connectors and switches associated with the transducers being measured.

ADC The adc has five measurement ranges. During the first scan, or as a result of an initialising scan (such as that used for strain gauge measurement), the range for each channel is automatically established and stored. On subsequent scans the same respective range for each channel will be adopted. Should an input be too high or too low for the adopted range, the adc will automatically move to the appropriate range. When an overload is detected, even if caused by a transient, the measurement is stopped and the adc up-ranges so that the input can be re-measured.

Selectors Each input in turn is connected to the measuring circuit by means of switches on a selector card. Two types of analogue selector are offered:

Reed Relay Selector (35301A) with reed relay switches
Solid State Selector (35301B) with solid-state MOSFET switches

ORION can be fitted with up to 10 selectors of either type. The Reed Relay type has the ability to handle high input and common mode voltages, whereas the Solid State type has faster switching (up to 500/s) and solid state reliability. N.B. Should any logger be fitted with a mixture of Reed Relay and Solid State Selectors the input and common mode voltages must be restricted to those specified for Solid State Selectors. This applies even if the Solid State channels are not in use. **It is therefore recommended that selector types are not mixed in any one logger.**

The switching organisation is the same for both types of selector. Both have a capacity of 20 channels of 3-poles, or 10 channels of 6-poles. Channels are numbered 1 through to 20, (and thereafter, in groups of 20, up to 200). If 6-pole inputs are used the channels adopt the odd numbers. Both 3-pole and 6-pole inputs may be intermixed in any order in any selector. No linking is necessary, all switching occurs automatically as a result of the input sensor definitions which are entered at the front panel.

Input Connectors The wiring from sensors, transducers and other inputs is taken to screw terminals located within a plug-in Input Connector (35303A/B), fitted to each selector. Each connector is totally enclosed and incorporates wiring clamps. Temperature sensors are fitted inside the connectors (35303A only); these give 'cold-junction' indication when the thermocouples inputs are connected.

Transducer energisation Certain types of transducer, such as strain gauges, need current energisation. This is provided automatically by ORION for the appropriate channels; the energising currents are automatically switched by the selectors and are output to the transducers via input connector terminals.

3.3 DIGITAL INPUTS

Both 'status' and 'event' inputs, and bcd or binary information can be connected to ORION. Single inputs may be derived from contacts, electronic switches and sensors. Bcd or binary information may come from other instruments and measurement modules.

An important distinction is made between status and event inputs. Status inputs are scanned sequentially and a voltage level at the input is detected. Event inputs are monitored continuously, and the transition of an input voltage from one level to another interrupts the processor as and when it occurs.

Two units are available for digital inputs. One is the Event/Status Input Module (35302A) which has 20 input pairs, any of which can be used for status, event or parallel bcd inputs. The other; the Digital Input/Output Module (35302B), has 20 inputs which can be used individually for status, or in groups for bcd and binary inputs. The Digital Input/Output Module additionally has 20 digital output lines. These outputs can be operated independently, or in conjunction with the inputs in order to read data from external modules such as counters and a-to-d converters.

Inputs are connected to the Event/Status Input Module using the 35303A Input Connector. Inputs (and outputs) are connected to the Digital Input/Output Module using the two 25-way connectors on the rear panel.

Status Status inputs can be scanned at speeds in excess of 500 channels/second, and can be used as individual channels or in groups. A group of channels (connected to 35302A) is regarded as one input, thereby saving time and record length when logging. Groups of 4 channels can read 8-4-2-1 bcd; one module can thus access 5 bcd decades from an external device such as a counter.

Events Event inputs, being continuously monitored, detect a logic-level transition when it occurs. The printed record of an event includes the time of occurrence. Any event channel can be used to totalise (count) a maximum of 65535 events. Successive readings may be either a cumulative total, or a new total from a zero reset on the previous reading. Using this facility a count of events can be produced, or alternatively, a conversion equation may be applied to the total (using the Conversion Definition) permitting it to be converted to user defined units.

Events can be processed at rates of up to 400/s; this defines the number of channels that can be processed at a particular event rate. Thus if a number of channels are used for events the handling speed is restricted to 400 events/s (total) for that group of channels.

A further optional Module, the Counter-Timer 35302C, permits any of the following four measurement functions to be performed:

Frequency measurement
Totalising
Period measurement
Interval measurement

The Module has five input channels which can be used independently for any of the measurement functions. The input signals may be differential or TTL compatible, and be Active-hi or Active-lo logic.

4 OUTPUTS

4.1 DIGITAL OUTPUTS

The optional Digital Input/Output Module (35302B) has 20 output lines. These can be used for alarm outputs, stimulus outputs, or, in conjunction with the inputs for collecting data from external modules.

Alarm outputs can be used to indicate an alarm state on any analogue or status input channel. Stimulus outputs are controlled by the processing on a scan task, individually or in groups. Grouped channels form bit patterns which can be set-up as required, and made to change at prescribed times or by triggers from other tasks. Thus, the outputs could be used to control a test rig through various stages of operation.

4.2 ANALOGUE OUTPUTS

The optional Analogue Output Module 35301E has 5 isolated output channels which may be independently configured for voltage or current output. The Module gives ORION a general-purpose analogue output capability which may, for example, be used to drive chart recorders, plotters or similar devices.

Each analogue output can be assigned to an input channel, using Channel Definition, or may be controlled by the processing on a scan task. When the input channel is scanned, the analogue output will assume an output level which is proportional to the measured input. The output can be scaled as required and will produce an output within the range:

0 V to 10 V for voltage or
4 mA to 20 mA for current.

5 SETTING-UP & CONTROL

ORION is set-up and controlled using the front panel keys in conjunction with the 2-line display. The fundamental parts of the set-up routine are Channel Definition and Task Definition, with any associated processing.

Channel Definition The Channel Definition key is used to define parameters for all channels. When the key is pressed the display asks which channels are to be defined; individual or grouped channels can be entered. The various steps in the setting-up procedure are presented in turn so that the parameters associated with channels may be entered. The parameters include type of measurement and sensor, alarm limits and action to be taken on alarm. The step-by-step requirements are defined either by selecting one of the six 'soft-keys' located underneath the display or by entering data using the keypad which contains all the alpha-numeric characters. If necessary the operator can go back, or forward, in the sequence by using the ↑↓ controls.

Certain parts of the setting-up are done automatically by the logger. For instance if analogue channel numbers are selected, event or status parameters are not offered. Similarly, incorrect allocation of channels is not permitted as ORION is able to detect the type of selector in each slot. Additionally, when a channel is defined for a particular configuration (say, 4-terminal resistance thermometer) ORION detects if energisation is required; and when that channel is measured, the correct switching is arranged automatically. See pages 9.6–9.11 for further details.

Task definition Tasks define the type and timing of logging and control routines, and decide whether processing shall occur or not. Up to 8 tasks can be defined. The higher numbers have priority over lower numbers. When the Task Definition key is pressed, menus are displayed from which selections are made using the soft keys, or data is entered from the keypad. Task Definition parameters include channel numbers, measurement speed and output device. The following information could be included in the Task Definition for a simple task: channels 1 to 5 to be measured at 40 channels/s, 20 scans at 10 minute intervals, the output to be shown on the printer. Limits could be applied to the channels and printing would be conditional on these limits. A scanning task can access analogue inputs and/or status inputs, both of which are

scanned by channel. Other tasks can observe event inputs or give stimulus outputs. See pages 9.12–9.15 for further details.

Starting a task A measurement task can be started at a specified time set using the logger clock, or it may be started after a delay which follows a RUN command. By using different clock times or different delays, several tasks can start independently. Alternatively, several tasks could be set-up to run concurrently.

A powerful feature of ORION is its ability to trigger one task from another. One task, which may be measuring analogue signals, or scanning status inputs or monitoring event inputs, can cause another task to start if a particular condition is found, such as an alarm. In addition any task can be triggered coincident with the start, or the end, of another task.

Using the powerful built-in facilities which ORION offers, the operator can readily set-up a comprehensive measurement and control program.

6 POWER SUPPLY & POWER FAILURE

ORION can be powered from 240V/115V ac mains or from an external (20 to 40V) battery. Battery operation permits use of ORION in vehicles, ships or aircraft; a 48Ah battery provides sufficient power for over 24 hours operation. Both mains and battery may be connected at the same time, in which case the battery would take over in the case of mains failure.

In the event of power failure, the internal clock and programs in the memory are maintained by an internal 'back-up' battery for at least 6 hours. This battery will be recharged when power is restored. The logger requires 4 days of operation to ensure a full 6-hour back-up capability if the battery is exhausted.

If ORION is switched OFF by the keyswitch on the front panel, the battery maintains the clock for several months; data in memory is automatically lost when the keyswitch is set to OFF, and the memory is therefore not maintained.

When 'power-failure' occurs all logging activity is shut down. When the power is restored the programmed routines are automatically restarted (providing failure does not exceed 6 hours). As an additional safeguard the user's program can be stored on the cartridge recorder. If power is then interrupted for longer than the internal battery will support the memory, the program is automatically reloaded from the cartridge when power is again available.

During a power failure the initialised conditions, such as unstrained parameters of strain gauges, are maintained in memory for up to 6 hours. The best safeguard against loss of these parameters is to SAVE the program after initialisation. If Program 1 is used, the program (including initialised conditions) will, if necessary, be automatically booted into memory after a power failure (assuming SW2/2 on the Controller is set to ON). This feature can be used to ensure the meaningful continuation of a program, or automatic loading and running of a program.

7 FRONT PANEL LAYOUT

The front panel layout is shown below in Fig 1.1. Full details on front panel control facilities may be found in Chapter 9.

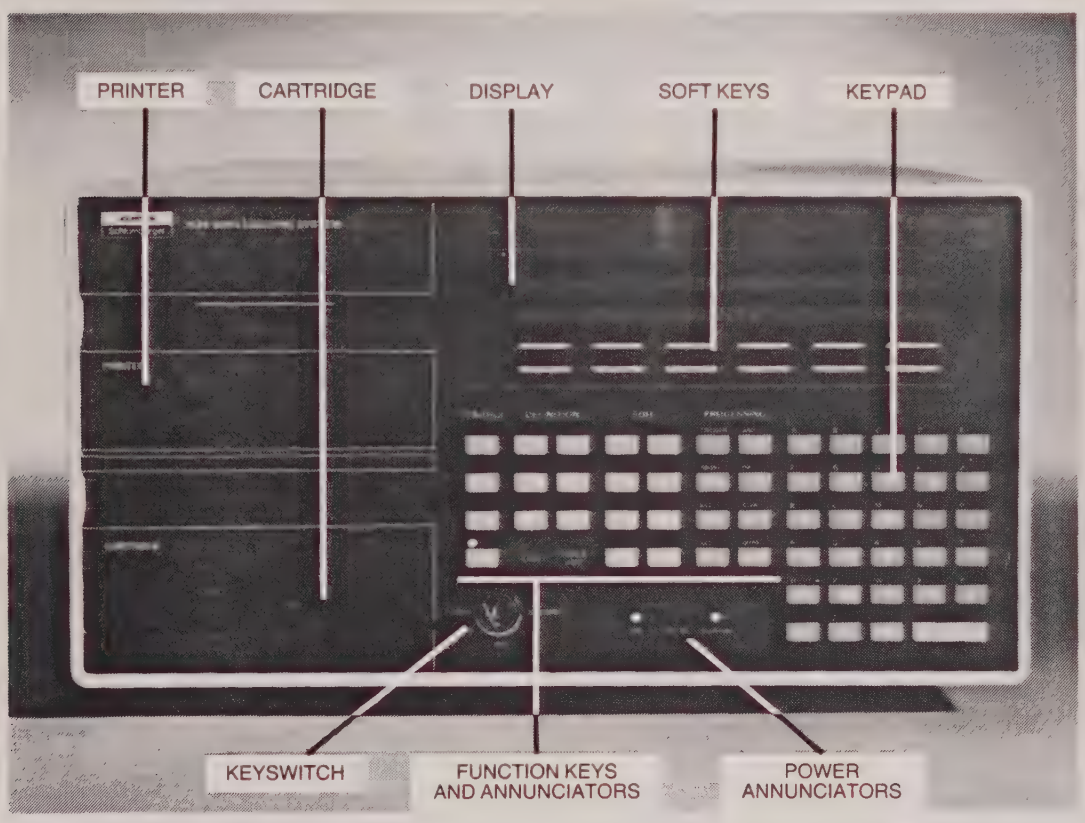


Fig 1.1 *ORION delta Front Panel*

8 OPTIONAL INPUT/OUTPUT MODULES

Each ORION has a 200 point input/output capability which can be configured to the user's requirements. This is achieved using the appropriate selection of optional plug-in modules. The full range of modules offered is listed below; each has a sales code number (e.g. 35301A) for easy reference when ordering.

Analogue Selectors

Reed Relay Selector 35301A
Solid State Selector 35301B

Each module has 20 input channels for any of the following types of input/sensor:
- Voltage ac or dc
- Current ac or dc
- Resistance
- Strain gauges
- PRT's
- Thermocouples

Analogue Output

Analogue Output Module 35301E: 5 output channels, each may be selected to give: current (4 mA to 20 mA, or voltage (0 V to 10 V).

Digital Selectors

Event-Status Input Module 35302A	20 input channels for status/event digital inputs.
Digital Input/Output Module 35302B	20 input channels for status digital inputs, and 20 output channels for alarm, stimulus or external module measurement.

Counter Timer Module 35302C : 5 input channels for:
- Frequency measurement
- Period measurement
- Totalising
- Interval measurement

Input Connectors

Input Connector 35303A	Fits 35301A and B, and 35302A.
Input Connector 35303B:	Fits 35301A and B; intended primarily for current inputs.

Other modules

Transducer Energisation Module 35301D	Generates energising currents or voltage for use with certain transducers.
GP-IB Interface 35304A	Permits connection of GP-IB-compatible (IEEE 488) instruments.
Serial Interface 35304B	Permits connection of RS422- or RS232-compatible instruments.

Rack Mounting Kit 35305D, see Chapter 3.

9 LOCATIONS OF STANDARD AND OPTIONAL MODULES

The plug-in standard and optional modules fit into slots at the rear of the logger, as shown in Fig. 1.2. The standard modules are shown shaded. Slot 2 and Slots 4 to 14 inclusive contain the optional modules.

Slot 2 accepts only the GP-IB or Serial Interface. Slot 14 accepts only the Transducer Energisation Module. Slots 4 to 13 will accept any of the analogue input or digital input or output modules specified, giving the logger its 200 point input/output capability. Apart from the undesirability of mixing analogue scanners (see Selectors, page 1.5) any mix of the specified modules is permitted; this allows modules to be interchanged if desired, or for additional modules to be added at a later date. Any vacant slots are fitted with blank panels.

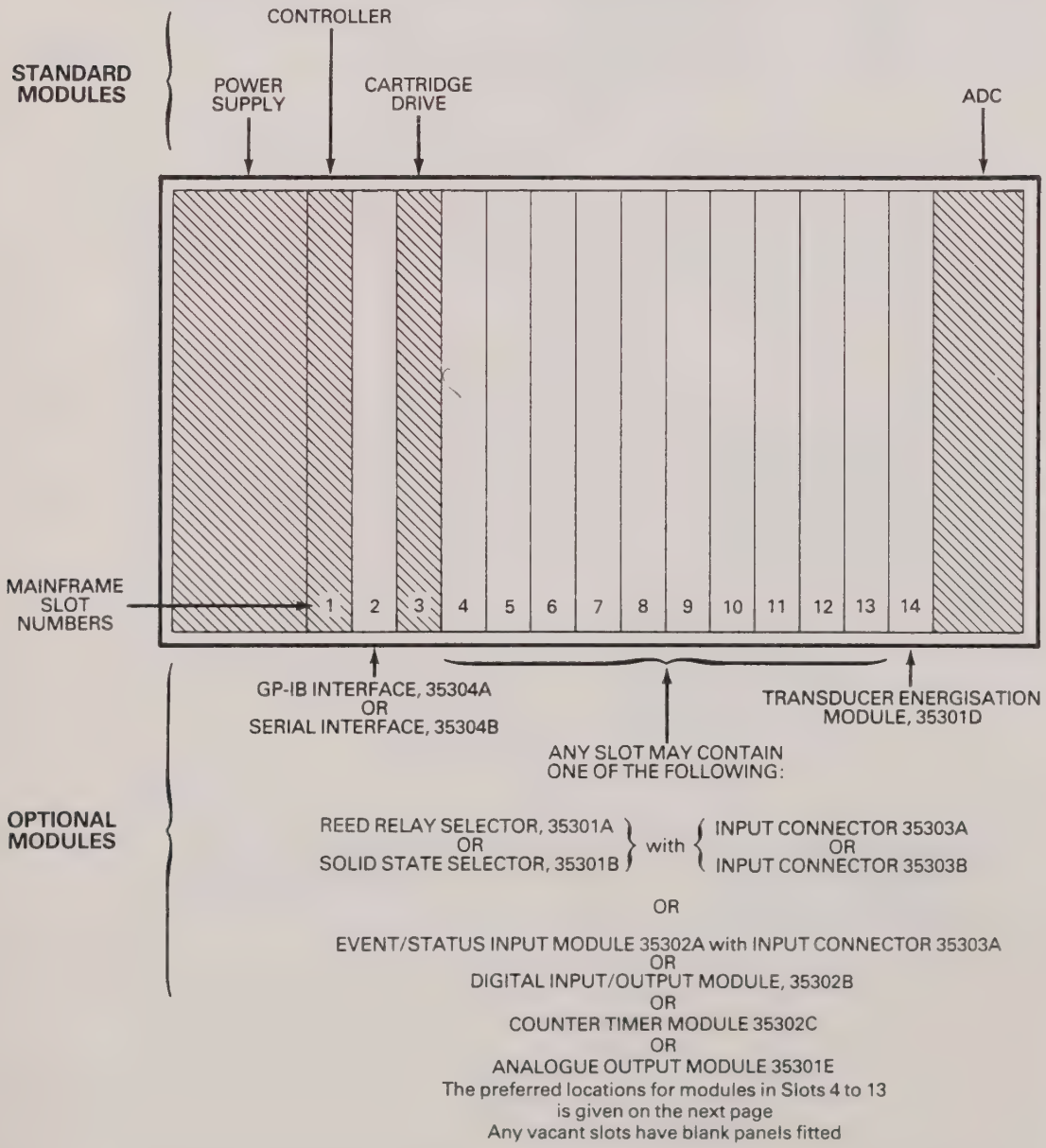


Fig. 1.2 *Rear view of ORION showing positions for standard and optional modules*

Preferred Locations for Slots 4 to 13

The Slots should be filled in the following order, starting at Slot 4:

Counter Timer Module	35302C
Analogue Output Module	35301E
Digital Input/Output Module	35302B
Event Status Input Module	35302A
Blanking Panel, with thermal barrier	35300019A
Reed Selector	35301A
Solid State Selector	35301B
Blanking Panel (1 unit long)	35300018A, or
Blanking Panel (4 units long)	35300020A

The rear view of ORION with a typical selection of modules fitted is shown in Fig. 1.3.

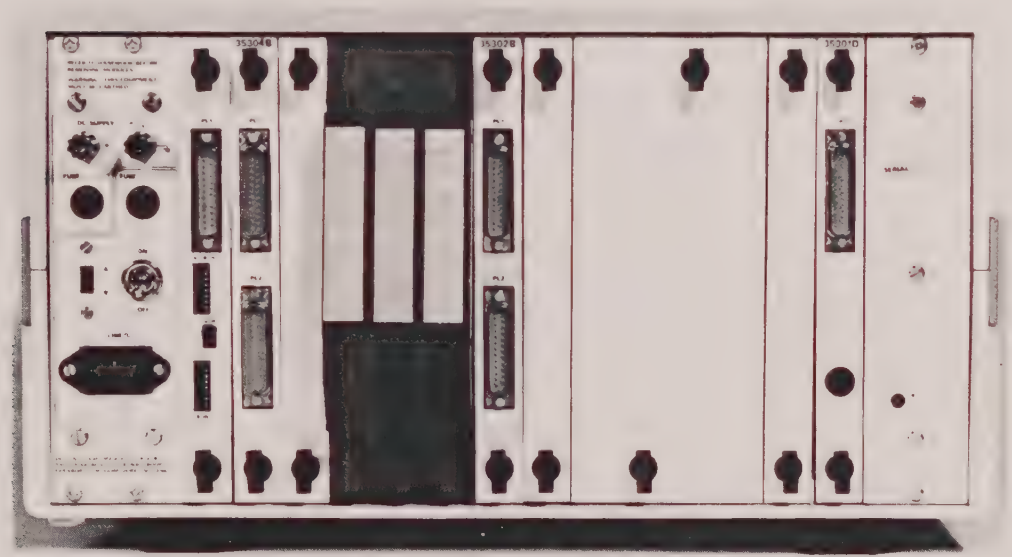


Fig. 1.3 *ORION with a typical selection of modules fitted.*

10 PERIPHERAL DEVICES

Any of the devices listed below may be used with ORION. Those listed are all supported by Solartron, other compatible devices may be equally suitable.

ADDS Regent 60 VDU: Silent 743 Printer: Silent 700: Trend 800 Printer: Facit 4208 Cassette Reader:	Any one may be connected to: Controller (PL1), or Serial Interface (PL1/PL2)
Any GP-IB controller device:	May be connected to: GP-IB Interface (PL1)

Chapter 2
Using ORION

Contents

This Operating Manual contains detailed information on the use and operation of ORION. Since ORION has a very flexible logging capability, it is not practicable to give specific operating procedures. Instead, the following instructions are offered.

1. **General Instructions for Using ORION** (see page 2.2)

This is a general guide for any application; it includes instructions for changing the paper roll in the printer, and the cartridge. In this section the user is referred to relevant Chapters in the manual where detailed information may be found on specific aspects of the logger.

2. **Example Setting-Up Procedure** (see page 2.5)

This is a very simple procedure intended solely to familiarise the new user with the control and operation of ORION. It is not a procedure for general use, and it only shows a very limited part of logger operation.

3. **Safety** (see page 2.9)

The user is advised to read these safety precautions before installing or using the logger.

1 GENERAL INSTRUCTIONS FOR USING ORION

1. If the user is unfamiliar with ORION, refer to CHAPTER 1 for general information.

2. Read the notes on SAFETY, see page 2.9.

3. Install ORION, and any associated instruments, as detailed in CHAPTER 3.

4. Prepare a 'logging schedule'; write down the logging routines and channel details as follows:

   (a) Define the type of all inputs and outputs, grouping similar types where practicable.
   (b) Define the logging 'Tasks'; remember that up to 8 Tasks can be run simultaneously, and they can be cross triggered if required.
   (c) Incorporate the channels into the Tasks.
   (d) Write the processing programs, as described in CHAPTER 12.

5. Connect all inputs and outputs.

   (a) For analogue inputs refer to CHAPTER 4 (use Reed Relay Selector 35301A or Solid State Selector 35301B)
   (b) For status/event inputs refer to CHAPTER 5 (use Event/Status Input Module 35302A)
   (c) For digital inputs and outputs refer to CHAPTER 6 (use Digital Input/Output Module 35302B)
   (d) For frequency, totalise, period and interval inputs refer to CHAPTER 7 (use Counter Timer Module 35302C)
   (e) For analogue outputs refer to CHAPTER 8 (use Analogue Output Module 35301E)

6. Switch-on as follows:

   (a) Set mains ON/OFF switch at rear to ON
   (b) Set Keyswitch on front panel to ON
   (c) Switch on any peripheral equipment (eg VDU) which is connected to ORION.

   The display will show a 'POWER FAIL' message indicating the time and date at switch-on. When the logger is switched on for the first time, or if the internal back-up battery has become discharged, the time and date indicated will be incorrect and should be reset using the INITIALISE key. Should a 'power failure' occur while the logger is running then, when the power is restored, a message is shown on the display indicating 'POWER FAIL AT (time and date)' and 'POWER BACK AT (time and date)'.

7. Enter the setting-up information as follows:

   (a) Use the front panel keys; refer to CHAPTER 9 for details on front panel controls. Refer also to your 'logging schedule'.
   or (b) Load the information from the cartridge if an appropriate program has been saved.

Fitting Paper Roll in Printer

When it becomes necessary to fit a new paper roll in the printer, carry out the following procedure:

(a) If logger is on and set to RUN, set PAUSE/CONTINUE to PAUSE (annunciator lit). See 'Use of PAUSE/CONTINUE facility' page 2.5.

(b) Pull down the hinged front cover on printer.

(c) Use the 'Lift-up to Release' lever inside the printer and pull out printer assembly (which is fitted on runners), see Fig 2.1.

(d) Lift out core of used paper roll and spindle.

(e) Fit spindle in new roll. Feed paper through aperture marked A in Fig 2.1 (with paper coming off front of roll). Drop roll and spindle into the roll compartment, locating the spindle in springs at either side.

(f) Feed paper up behind rubber roller. Push paper up, and at the same time press Paper Advance Key. Paper will then appear between roller and print head.

(g) Press Paper Advance key, until there is a length of paper about 6'' hanging from the front of the mechanism. Push mechanism back into the logger.

(h) Feed paper through aperture marked B in Fig 2.1 in hinged front cover and swing cover back into position so as to conceal printer mechanism.

(i) Set PAUSE/CONTINUE to CONTINUE (if applicable).

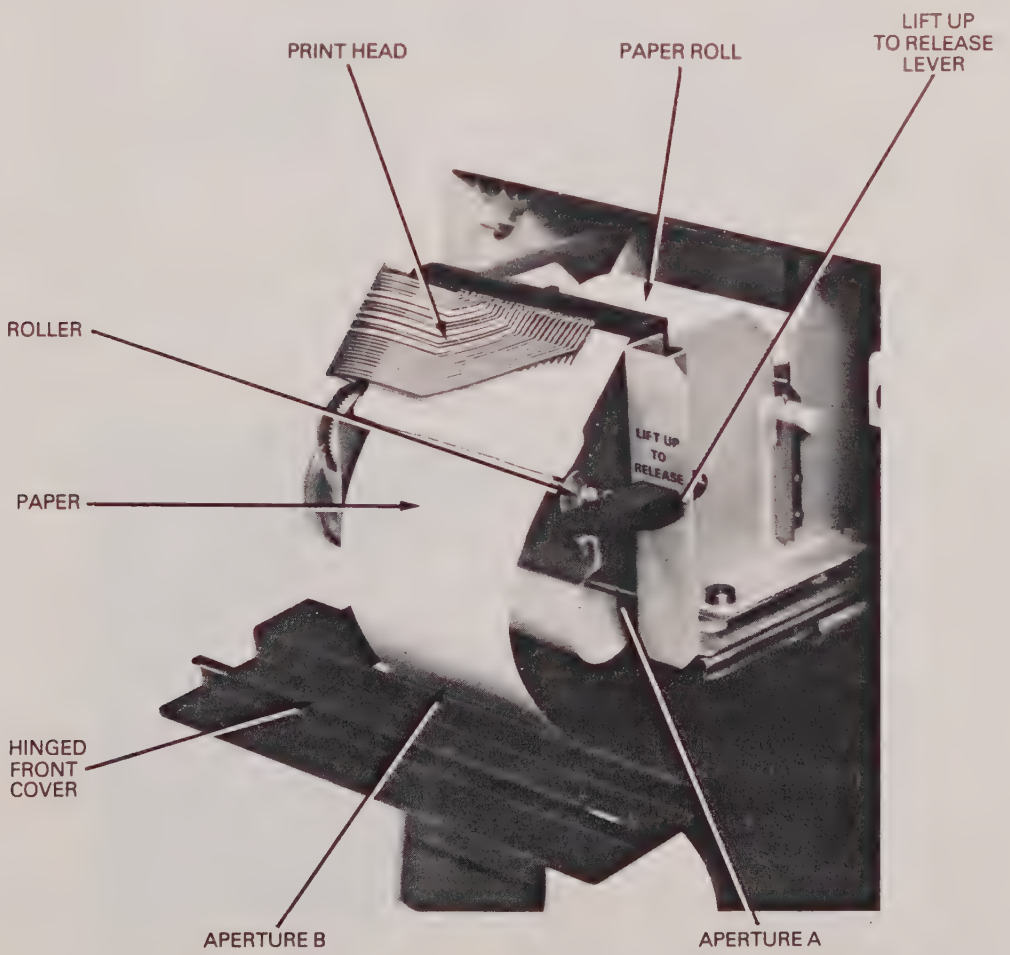


Fig 2.1 *Replacing paper roll in printer*

Ejecting and fitting a cartridge

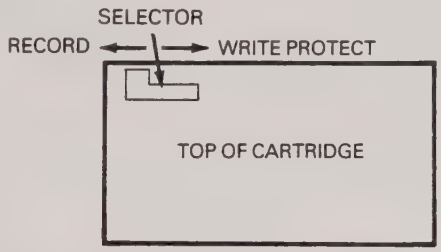
The cartridge may be changed only if:
the logger is in the HALT mode (the tape may be in any position),

or

the logger is in the RUN mode, and the logger itself has initiated the PAUSE state as a result of the tape being completely filled.

To eject or fit a cartridge carry out the following procedure (note that only type DC100A cartridges can be fitted):

(a) Pull down hinged front cover on cartridge recorder.
(b) To eject a cartridge, press the Eject button, see Fig. 2.2. and withdraw cartridge.
(c) Before fitting new cartridge ensure that the Record/Write-Protect selector on cartridge is set as required, see below.



RECORD: Record and replay functions can be used.

Record functions:
Program MARKER
SAVE program
Log data
ERASE data

Replay functions:
DIRECTORY
LOAD programs
COPY logged data

WRITE PROTECT: Replay functions only

Replay functions:
DIRECTORY
LOAD program
COPY logged data

(d) Insert cartridge in aperture (top side uppermost) until it locates.
(e) Set PAUSE/CONTINUE to CONTINUE (if applicable)

The cartridge will automatically rewind to correct operating position.

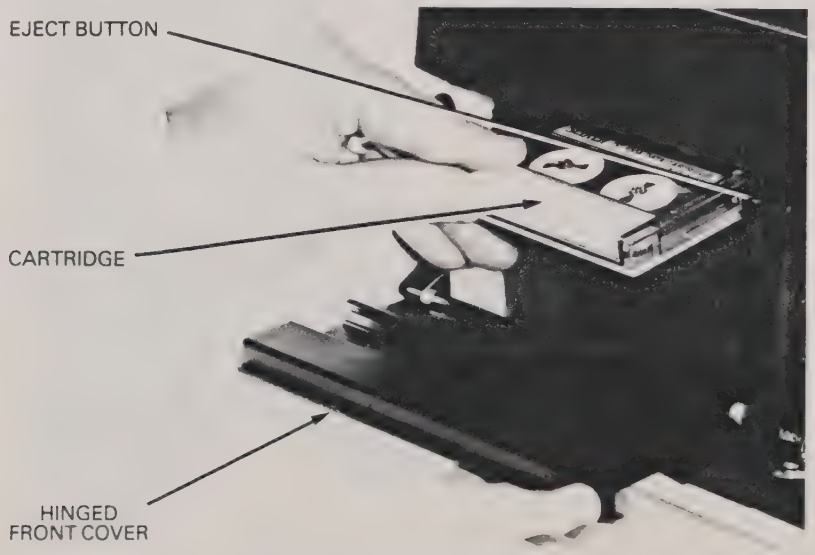


Fig. 2.2 *Inserting a cartridge*

Use of PAUSE/CONTINUE facility

The PAUSE/CONTINUE facility is intended for use while changing the paper roll when the logger is set to RUN.

When set to PAUSE, all programmed logging tasks proceed normally except that no logged data is output to the printer. Instead, any data logged is stored in an internal buffer (which holds up to 1000 measurements). The paper roll can then be changed without interrupting logging routines and without the possibility of data being output to the printer. After changing the paper roll the logger is set to CONTINUE. This will cause any data logged (and stored in the buffer) during the PAUSE period to be output to the printer.

The PAUSE/CONTINUE facility thus permits the results of logged measurements to be temporarily stopped, without interfering with logging routines and without loss of measured data. The facility is intended only for temporary use; the internal buffer can store only 1000 measurements, and when it is full, logging will automatically stop. When this happens, logging will not restart until CONTINUE has been selected. Subsequent task triggers may then be delayed. It is therefore not advisable to use PAUSE/CONTINUE when the logger is expected to make more than 1000 measurements during the PAUSE period.

2 EXAMPLE SETTING-UP PROCEDURE

2.1 INTRODUCTION

An example of a simple setting-up procedure is given below. The example is intended purely to familiarise the user with ORION, and should not be regarded as a procedure for general use.

The example procedure can be summarised as follows:

(a) Channel Definitions for channels 1-5 are entered.
(b) A simple scanning Task, incorporating channels 1-5, is defined.

The procedure will program the logger to scan channels 1-5 and output the logged measurements at the printer. It is not neccessary to connect any inputs to channels 1-5 as the logger will measure and record the open-circuit voltages present on these channels, this being sufficient to demonstrate operation of the logger. Note that the Input Connector must be fitted.

Before carrying out the 'Example Procedure' connect the mains (or battery) supply as described in CHAPTER 3, observing the safety requirements on page 2.9.

The procedure assumes that channels 1-5 are analogue channels, i.e. a Reed Relay Selector 35301A or a Solid State Selector 35301B is fitted to slot 4 at rear of logger. If neither type is fitted to slot 4, the channel numbers entered in instruction (b) should be changed to correspond with analogue channels elsewhere in the logger, see page 4.9 for channel numbering details.

2.2 EXAMPLE PROCEDURE

1. Set Keyswitch on front panel to ON.
2. Ensure that the RUN key is set to HALT (annunciator not lit).
3. Set-up the Channel Definition as shown in (a) to (e) below, pressing the keys indicated under 'ACTION'. Note that any key shown thus: Sensor refers to a 'soft key', e.g. the key immediately below the word 'Sensor' on display. (See Fig 1.1 page 1.9 for general location of front panel controls). The phrases below in italics will appear on the display; each will appear automatically after the previous key selection has been made.

FUNCTION	ACTION
(a) Select Channel Definition	CHAN
(b) *Enter channel numbers*	P 1, N –, L 5, ENTER
(c) *Select item to be defined*	Sensor
(d) *Select measurement type*	Volts
(e) *Select DC or AC measurement*	DC

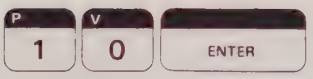
4. Set up the Task Definition as shown in (a) to (s) below, pressing the key indicated.

FUNCTION	ACTION
(a) Select Task Definition	TASK
(b) *Enter task number*	P 1, ENTER
(c) *Enter task title*	SHIFT, S /, C =, A <, N – SPACE, C =, H 9, A <, N –, S / SHIFT, SPACE, P 1, N –, L 5, ENTER
(d) *Select task function*	Scan

(e) *Select task trigger source*

Timer

(f) *Enter delay to start*



(= 10 s)

(g) *Enter number of scans*

(h) *Select continuous or fixed interval scan*

Inter

(i) *Enter Scan Interval*

(= 30 s)

(j) *Enter channels to be scanned*

(k) *Select measurement rate*

40/s

(l) *Select end-of-scan processing*

Off

(m) *Select data to be logged*

ALL

(n) *Select when to output task header*

Always

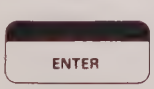
(o) *Select output device*

Print

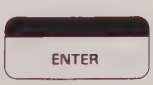
(p) *Select format required*

Full

(q) *Enter tasks triggered on Alarm*



(r) *Enter tasks aborted on Alarm*



(s) *Ready for command*

The Channel Definition has defined channels 1-5 as inputs for dc voltage measurement. These channels have been incorporated into Task 1, which has itself been defined as a 'scanning' task.

When 'RUN' is selected logging activity will start. After the defined delay time of 10 seconds (delay to start) Task 1 will produce a scan of channels 1-5. The scan will be repeated five times at 30 second intervals (scan interval), then scanning will stop. The display will show the header DT1 (done task 1). At each scan, the dc voltage input at each channel will be measured. The measurements will be output to the printer and will appear on the paper-roll print-out, see Fig 2.3. The measurements will be made at a rate of 40/s and be in full format, as requested in the Task Definition.

5. Press RUN key to select RUN (annunciator lit).
6. To stop Task, press RUN again to select HALT.

The setting-up procedures for more extensive and comprehensive logging routines are carried out in a similar fashion.

Having successfully set-up and 'run' a simple Task the user is again reminded that up to 8 'Task Definitions' can be set-up. Channel Definitions would, in practice, be set-up for all input channels to be used, and these channel numbers would then be incorporated into the appropriate Tasks. Any specific channel(s) may be incorporated in any number of Tasks.

Tasks are divided into 2 types: Scanning and Event. Each Task can be set-up as any type. Regardless of type selected, the Task can be programmed to run independently, or as an interdependent (cross-triggered) Task. In this case, any Task can be triggered by any other.

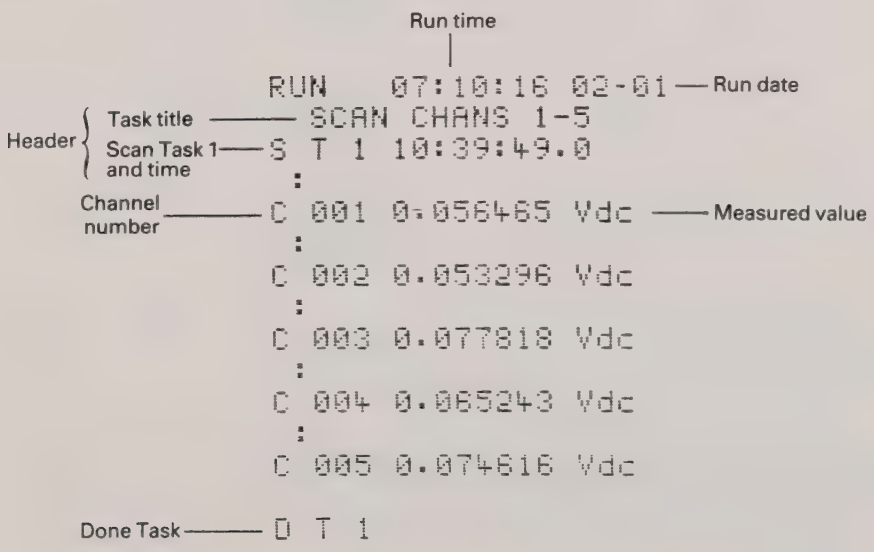


Fig 2.3 *Typical print-out produced by Task 1*

3 SAFETY

The 3530 ORION data logger has been designed in accordance with IEC publication 348 (Class 1), Safety Requirement for Electronic Measuring Apparatus, and has been supplied in a safe condition. This operating manual contains information and warnings which must be followed by the user to ensure safe operation and to retain the equipment in a safe condition.

The logger has been designed for indoor use, but may occasionally be subjected to temperatures between +50°C and −10°C without degradation of its safety.

This manual gives operating instructions together with any safety precautions where appropriate. The principle safety precautions are also listed below.

Safety Precautions

1. Before switching on the logger ensure that the mains voltage selector is correctly set.

2. Ensure that the mains plug is connected only to a mains outlet which has a protective earth contact. This applies equally if an extension lead is used, the lead must contain an earth conductor.

3. To effect earthing, the mains plug must be inserted before connections are made to measuring and control circuits. If a dc power source only is used an external earth must be connected before connections are made to measuring and control circuits. The mains plug or external earth (as appropriate) must remain connected until all measuring and control circuits have been disconnected.

4. Any interruption of the earth connection (inside or outside the logger) is prohibited.

5. When the logger is connected to its supply the opening of covers or removal of parts could expose live conductors. The logger should be disconnected from all voltage sources before it is opened for any adjustment, replacement, maintainance or repair. Adjustments, maintainance or repair of the logger when it is powered should not be attempted by the user. Consult a Solartron Service Centre if repairs are necessary.

6. Ensure that only fuses of the correct rating and of the specified type are fitted. Makeshift fuses, and short-circuiting of fuse holders is prohibited.

7. Whenever it is likely that the protection of the logger has been impaired, it should be made inoperative and secured against any unintended operation. Protection could be impaired if the logger:

   i) shows visible damage
   ii) fails to perform the intended measurements
   iii) has been subjected to prolonged storage under unfavourable conditions.
   iv) has been subjected to severe transport stress.

8. The ⚠ symbol on the logger means 'Refer to Operating Manual' for detailed instructions or safety precautions.

Chapter 3
Installation

Contents

This chapter contains details for connecting the ac or dc supply, connecting peripheral devices (if any) to the logger and for rack mounting. Peripheral devices may be connected to certain modules in the logger; the connection details are sub-divided into the different module types (sections 2.2-2.4). The information in this chapter is presented as shown below.

		Page No.
1	Connecting the AC mains or DC supply	3.2
1.1	Safety precautions	3.2
1.2	General	3.2
1.3	Procedure	3.2
2	Connecting Peripheral Devices	3.4
2.1	General	3.4
2.2	Controller pcb	3.5
2.3	GP-IB Interface (35304A)	3.7
2.4	Serial Interface (35304B)	3.10
3	Rack Mounting	3.13
4	Slave Expansion Unit 3530E	3.17

1 CONNECTING THE AC MAINS OR DC SUPPLY

1.1 SAFETY PRECAUTIONS

Earthing For reasons of safety an earth connection is essential whenever measurement and control circuits are connected, even if the logger is switched off. Earthing is achieved by connecting the logger to a mains outlet, or if using a dc supply only, by connecting the DC SOURCE -ve terminal to a suitable earthing point. This earth should be capable of carrying 25A and conform to the regulations in 'British Standard Code of Practice CP1013 1965, Earthing'.

See also the general notes on SAFETY in CHAPTER 2, page 2.9.

1.2 GENERAL

The logger may be powered from 115V/230V ac mains or from a 20V to 40V dc source (such as a 24V battery having a capacity of at least 48Ah). One or both sources may be connected. If both are connected the logger automatically selects the source from which it derives power. If one source fails, or its voltage level becomes too low, power will be derived from the other source. The point of changeover will depend on load conditions in the logger, and relative voltage levels of the supplies.

1.3 PROCEDURE

1. Set the voltage selector on Power Supply to 115V or 230V as appropriate, see Fig 3.1. Ensure that mains fuse rating is correct; 1A SLO BLO for 230V, 2A SLO BLO for 115V.

2. Ensure that the mains switch at rear is set to OFF.

3. If using ac mains supply, connect the mains lead.

4. If using the dc source, connect the dc supply to the DC SOURCE terminals, observing correct polarity.

5. If dc source only is connected, connect an earth lead between the DC SOURCE -ve terminal and a suitable earthing point.

When the logger is on the POWER annunciators on the front panel indicate the state of the power sources, (see 'POWER annunciators Chapter 9).

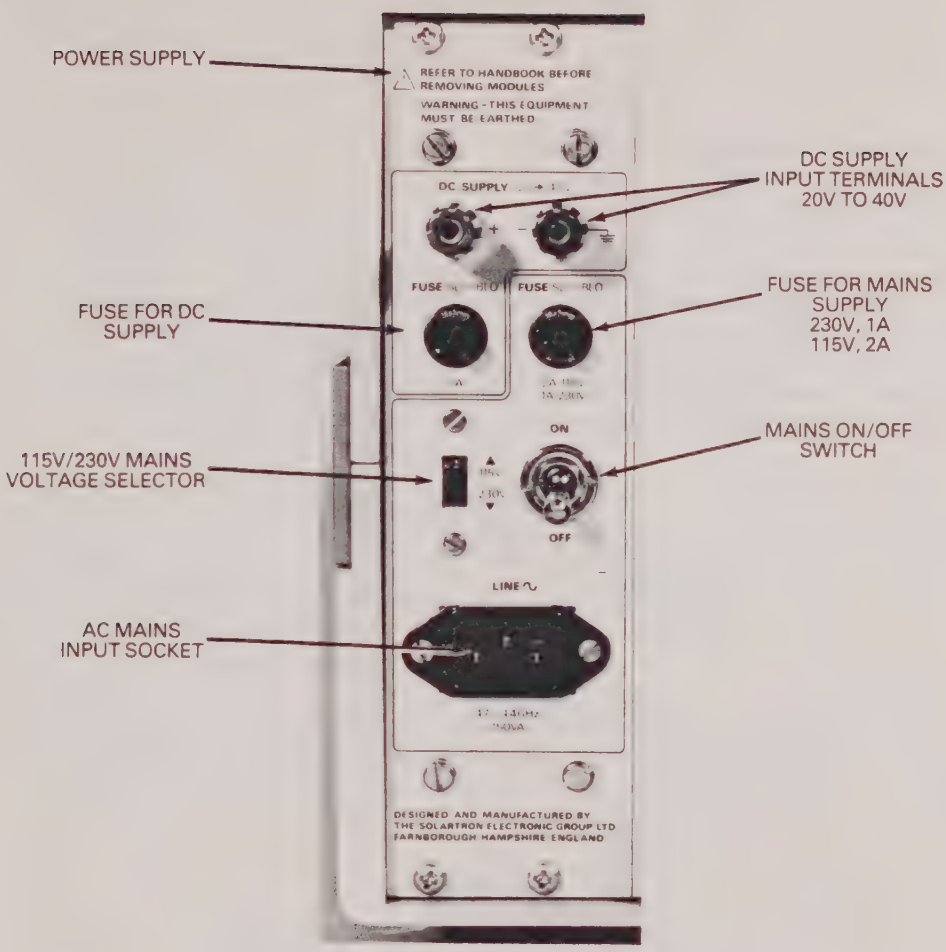

POWER SUPPLY
REFER TO HANDBOOK BEFORE REMOVING MODULES
WARNING - THIS EQUIPMENT MUST BE EARTHED
DC SUPPLY
DC SUPPLY INPUT TERMINALS 20V TO 40V
FUSE FOR DC SUPPLY
FUSE FOR MAINS SUPPLY 230V, 1A 115V, 2A
ON
MAINS ON/OFF SWITCH
115V/230V MAINS VOLTAGE SELECTOR
OFF
LINE
AC MAINS INPUT SOCKET
DESIGNED AND MANUFACTURED BY THE SOLARTRON ELECTRONIC GROUP LTD FARNBOROUGH HAMPSHIRE ENGLAND

Fig 3.1 *Power Supply*

2 CONNECTING PERIPHERAL DEVICES

2.1 GENERAL

Peripheral devices may be connected to the logger using the appropriate plugs/sockets provided on certain logger modules (mating connectors are supplied). The location of the plugs/sockets is shown in Fig 3.2. Before connecting a peripheral device, refer to the appropriate section of this chapter as indicated in the table below. The Peripheral Devices listed in the table are typical, other compatible devices may be equally suitable.

MODULE	OUTLET	TYPICAL PERIPHERAL DEVICES	SECTION
CONTROLLER	PL1	Any RS232C compatible device such as: SILENT 743 PRINTER SILENT 700 TREND 800 PRINTER FACIT 4208 CASSETTE RECORDER VDU	2.2 (see page 3.5)
GP-IB (IEEE 488)	PL1	Any GP-IB compatible device such as: Process controller or calculator (talker/listener) Output peripherals (listener only)	2.3 (see page 3.7)
SERIAL INTERFACE	PL1(port a) PL2(port b)	Any RS422/232C compatible device such as: SILENT 743 PRINTER SILENT 700 TREND 800 PRINTER FACIT 4208 CASSETTE RECORDER	2.4 (see page 3.10)
	(Port a only)	PENNY & GILES 7000 CARTRIDGE RECORDER	

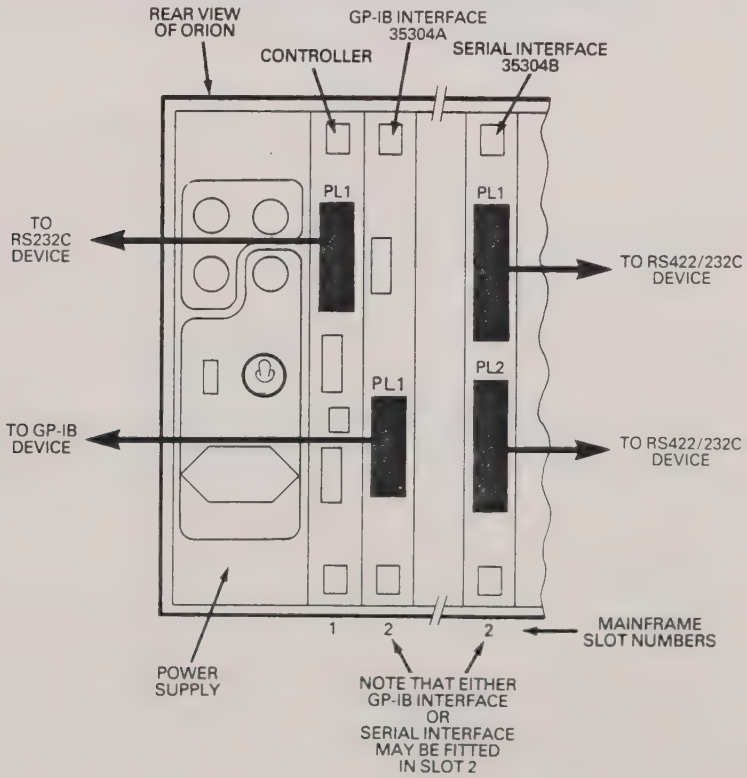


Fig 3.2 *Location of plugs/sockets to which peripherals may be connected*

2.2 CONTROLLER PCB

The following components are located on the rear panel of the Controller (see Fig 3.3):

Connector PL1:	for test purposes, and for connecting external RS232C-compatible devices; the RS232C standard is described briefly below.
Switches SW1 and SW2:	For selecting various functions including baud rate and external device type, see 'SW1 and SW2 Switching Settings' below.
Abort switch:	for test use.

RS232C This standard defines a single-ended bi-polar voltage circuit, capable of data rates up to 20 kilobauds (the Controller will handle data rates up to 9.6 kB). The maximum length of conductor is 50 feet. This single-ended circuit uses a single conductor to carry each signal, with the voltage referenced to a common return conductor.

The Controller does not implement a full subset of RS232C but nevertheless the facility offered is sufficient for most RS232C applications, see the PL1 pin details table on page 3.6.

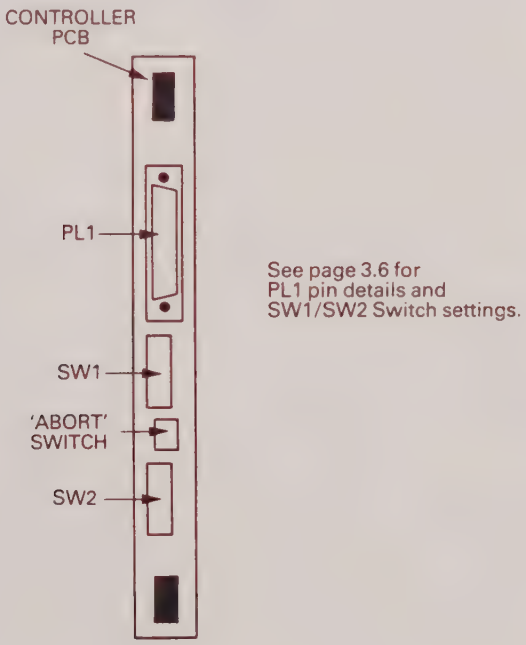
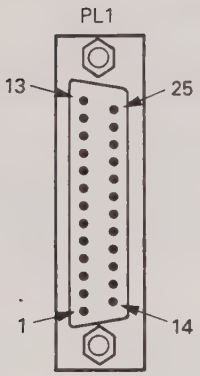


Fig 3.3 *Location of PL1 and switches on Controller*

PL1 Pin Details

PIN NO	SIGNAL
2	TX DATA
3	RX DATA
4	RTS (REQUEST TO SEND)
5	CTS (CLEAR TO SEND)
7	SIGNAL GROUND
20	DTR (DATA TERMINAL READY)

PLI carries RS232C signals permitting any RS232C compatible device (e.g. Silent 700, Regent 60) to be connected.

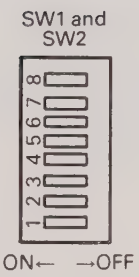


SW1 Switch Settings

SWITCH CONTACTS	FUNCTION	SWITCH POSITION OFF	SWITCH POSITION ON
1	SELECT BAUD RATE	SET ALL CONTACTS TO 'OFF' EXCEPT FOR THE ONE BAUD RATE SELECTED	75/110B*
2			150B
3			300B
4			600B
5			1.2kB
6			2.4kB
7			4.8kB
8			9.6kB

* DEPENDENT ON SETTING OF SW2/1

Note orientation of SW1 and SW2: Contact 8 is located at top of switch, contact 1 at the bottom.



SW2 Switch Settings

SWITCH NUMBER	FUNCTION	SWITCH POSITION OFF	SWITCH POSITION ON
1	SELECT LOW BAUD RATE (SEE SW1/8)	75B	110B
2	PROGRAM BOOT ON/OFF *SEE BELOW	OFF	ON
3	SELECT NORMAL OR TEST	NORMAL	TEST
4	SELECT FORMAT AND PROTOCOL FOR DEVICE CONNECTED TO PL1 ON CONTROLLER	**SEE DETAILS IN ADJACENT COLUMN	
5			
6			
7	NOT USED	–	–
8	WATCHDOG	OFF	ON

*PROGRAM BOOT ON/OFF For program booting after Power Failure

ON: If memory is corrupt: the logger attempts to load Program 1 from the cartridge. If this succeeds the logger runs Program 1, if it fails the logger clears program in memory.

If memory is OK: if the logger was in RUN at Power Failure the program in memory is re-run, if it was in HALT the program in memory is retained.

OFF: If memory is corrupt: the logger clears program in memory.

If memory is OK: the logger leaves program in memory intact, and logger assumes the HALT condition regardless of whether set to RUN or HALT before power failure.

**STANDARD SETTINGS FOR SWITCHES 4, 5 & 6

FUNCTION	SWITCH SETTINGS *As marked on rear panel SW2/4 S3*	SW2/5 S2*	SW2/6 S1*
PRINTING TERMINAL e.g. Silent 700 ASR, DECwriter teletypes and printer-only terminals. Provides hard copy output to low-speed printers, and permits control of logger from a terminal with keyboard. The output lines are terminated by CR LF and 4 nulls.	OFF	OFF	ON
VDU e.g. Any suitable VDU; the 80th character is used for CR. The VDU can be used to control logger, and to display data. The output lines are terminated by CR LF (if line length is exactly 80 characters LF is omitted).	OFF	ON	OFF
TAPE or PUNCH e.g. manually set up Silent 700, Facit tape cassettes, or paper tape punch. For output only, to external data recorder. Output lines are terminated by CR only.	ON	OFF	OFF
COMPUTER e.g. Apple with California Computer Serial I-O, HP calculator or computer with minimal interface. Hardware handshake not required, but 1 character buffering in hardware is required (e.g. an ACIA). The output lines are terminated by CR only. Before a string is printed the logger sends one character (O, R or E) to signify data ready, and waits for characters from computer indicating 'go ahead'.	ON	ON	OFF

2.3 GP-IB INTERFACE 35304A

Any GP-IB (General Purpose Interface Bus) instrument may be connected to PL1 on the GP-IB Interface 35304A, see Fig 3.4. The inputs and outputs at PL1 are compatible with the GP-IB IEEE 488 standard, which is briefly described below.

GP-IB standard This permits up to 15 instruments to be connected to any one Bus. Total bus length must not exceed 20 metres. Each instrument is designated as one of the following: CONTROLLER or TALKER/LISTENER.

(a) A CONTROLLER can transmit and receive bus commands, and can designate one device to be a TALKER, or several devices to be LISTENERS.
(b) A TALKER is a device transmitting data.
(c) A LISTENER is a device receiving data.

At any instant a bus can have only one active TALKER but may have several LISTENERS.

The bus consists of 16 lines, these are subdivided as follows:

5 lines for bus MANAGEMENT
3 lines for VALIDATION of information transfer (handshake)
8 lines for DATA, these may carry information or instructions as defined by the MANAGEMENT lines

MANAGEMENT lines

Interface Clear (IFC)
Remote Enable (REN)
Attention (ATN)
End or Identify (EOI)
Service Request (SRQ)

VALIDATION lines

Data Valid (DAV)
Not Ready for Data (NRFD)
Not Data Accepted (NDAC)

DATA LINES

DIO 1 to DIO 8 (DIO1 = l s b)

Using the GP-IB Interface, the logger can be designated as a talker/listener or a talker only. The talker/listener status and the address number for this module is selected using SW1, see page 3.9.

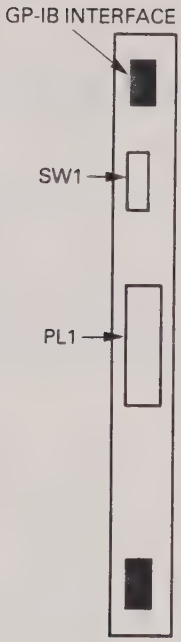
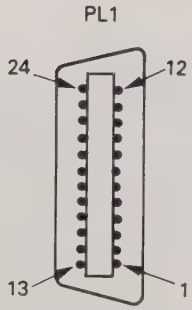


Fig 3.4 *Location of PL1 and SW1 on GP-IB Interface*

PL1 Pin Details

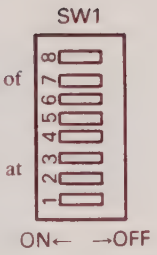
PIN NO	SIGNAL
1	DIO1
2	DIO2
3	DIO3
4	DIO4
5	EOI
6	DAV
7	NRFD
8	NDAC
9	IFC
10	SRQ
11	ATN
12	
13	DIO5
14	DIO6
15	DIO7
16	DIO8
17	REN
18	0V
19	0V
20	0V
21	0V
22	0V
23	0V
24	0V



SW 1 Switch Settings

SWITCH CONTACTS	FUNCTION	SWITCH POSITION OFF	SWITCH POSITION ON
1 2 3 4 5	TALKER/LISTENER ADDRESS	ADDRESS 0 TO 31, SEE TABLE BELOW	
6	TALKER/LISTENER STATUS	TALKER ONLY	TALKER/ LISTENER
7	NOT USED	-	-
8	NOT USED	-	-

Note orientation of SW1 and SW2: Contact 8 is located at top of switch, contact 1 at the bottom.



	ADDRESS																													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
SW1/1	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF	ON	OFF
SW1/2	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF	ON	ON	OFF	OFF
SW1/3	ON	ON	ON	ON	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF	ON	ON	ON	ON	OFF	OFF	OFF	OFF
SW1/4	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF
SW1/5	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	ON	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF

2.4 SERIAL INTERFACE 35304B

The Serial Interface 35304B has two output ports (a and b) which can be set independently for either RS422 or RS232C (limited specification) operation. PL1 = port a, PL2 = port b, see Fig 3.5. Selection of RS422 or RS232C standard at port a and b is achieved using SW4/1 and SW4/2, see Fig 3.6.

Protocol information can be found in the associated publication 'Communicating with ORION *delta*'.

Both RS422 and RS232C define standards for interfacing between data terminals and data communications equipment. The two standards are described briefly below.

RS422 This defines balanced-voltage differential operation, capable of data rates of 100 kilobaud over 4000 feet, or up to 10 megabauds over shorter distances. Since the system is balanced, it is possible to achieve very good immunity to common mode interference. Note that the Serial Interface (35304B) fully implements RS422 (the electrical standard) but does not implement RS449 (the functional and mechanical standard).

RS232C This defines a single-ended bi-polar voltage circuit, capable of data rates up to 20 kilobauds. The maximum length of conductor is 50 feet. This single-ended circuit uses a single conductor to carry each signal, with the voltage referenced to a common return conductor. The Serial Interface (35304B) does not implement a full subset of RS232C but nevertheless the facility offered is sufficient for most RS232C applications.

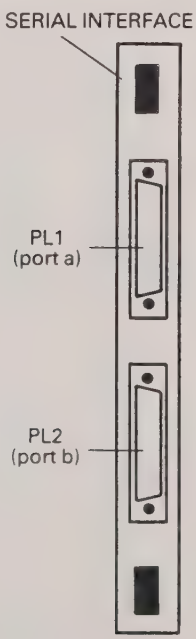


Fig 3.5 *Location of PL1 and PL2 on Serial Interface*

PL1/PL2 pin details

PL1 and PL2 PIN NUMBERS	RS232 SIGNALS	RS422 SIGNALS
1	GND	GND
2	Tx	
3	Rx	Rx Hi (balanced)
4	RTS (REQUEST TO SEND)	
7	0V	Rx Lo (balanced)
11		Tx Lo (balanced)
18		Tx Hi (balanced)
20	DTR (DATA TERMINAL READY)	

PL1 and PL2 are intended primarily for use by a controlling device. PL1, in addition, offers the protocol necessary for driving a Penny & Giles series 7000 data recorder (variant D17591/2R). PL2 can be used as an output only, with a choice of delimiting characters.

Only PL1 can be programmed from the front panel (using Task Definition, Output Device, '422a', see Chapter 9). PL2 can be programmed only from a remote controlling device, using the command 'TAn To S2', after setting up TAsk n in the normal way (see the associated publication 'Communicating with Orion' for further information).

When using the Penny & Giles data recorder, the error codes will be displayed as received. Reference should be made to the Penny & Giles manual for interpretation of the error codes.

Switches SW1-4

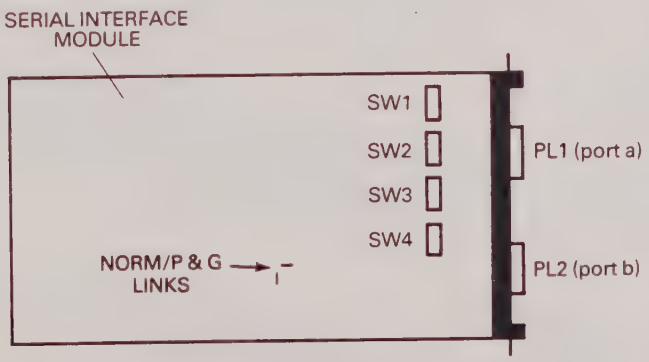


Fig. 3.6 *Location of Switches SW1-SW4 and Norm/P & G Links*

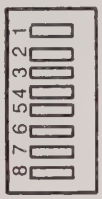
The switches SW1-SW4 permit selection of baud rate, parity and operating mode (RS232C/422) for the two ports, see tables on page 3.12.

If the Penny and Giles recorder is to be connected (to port A) connect the 'P & G' link (SP2) and open-circuit the 'NORM' link (SP1). For all other applications, connect the 'NORM' link (SP1) and open-circuit the 'P & G' link (SP2).

Switch Settings for SW1-4

SW1			
SWITCH CONTACTS	FUNCTION	SWITCH POSITION OFF	ON
1 2 3 4 5 6 7 8	SELECT BAUD RATE FOR PORT a (Continued on SW2)	SET ALL CONTACTS TO 'OFF' EXCEPT THE ONE BAUD RATE SELECTED	153.6kB* 76.8kB* 38.4kB* 19.2kB* 9.6kB 4.8kB 2.4kB 1.2kB
SW2			
1 2	SELECT BAUD RATE FOR PORT a (Continued from SW1)	SET ALL CONTACTS TO 'OFF' EXCEPT THE ONE BAUD RATE SELECTED	600B 300B
3 4 5 6 7 8	SELECT BAUD RATE FOR PORT b (Continued on SW3)	SET ALL CONTACTS TO 'OFF' EXCEPT THE ONE SELECTED	153.6kB* 76.8kB* 38.4kB* 19.2kB* 9.6kB 4.8kB
SW3			
1 2 3 4	SELECT BAUD RATE FOR PORT b (Continued from SW2)	SET ALL CONTACTS TO 'OFF' EXCEPT THE ONE BAUD RATE SELECTED	2.4kB 1.2kB 600B 300B
5	SELECT PARITY FOR PORT a (SW3/6 overrides SW3/5)	EVEN	ODD
6		NO PARITY	PARITY
7	SELECT PARITY FOR PORT b (SW3/8) overrides SW3/7)	EVEN	ODD
8		NO PARITY	PARITY
SW4			
1	SELECT RS232C/422 FOR PORT a	RS422	RS232C
2	SELECT RS232C/422 FOR PORT b	RS422	RS232C
3	SELECT PORT a PROTOCOL	(SW4/4 enabled)	BS4505 (overrides SW4/4)
4			Penny & Giles protocol (see Fig. 3.6)
5	SELECT PORT b PROTOCOL	(SW4/6 enabled)	BS4505 (overrides SW4/6)
6		C/R	C/R, L/F and 4 NULLS

SW1-3:
SW4 is similar, but has only 6 switch sections



*When using baud rates of 19.2kB and higher: The overall throughput is limited by software execution time rather than baud rate.

3 RACK MOUNTING

ORION can be rack-mounted, if required, in a 19'' wide rack of suitable depth. There are two ways in which ORION can be fitted in a rack.

1. Using the Rack Mounting Brackets provided in the standard accessories kit, together with additional support at the rear of the logger provided by the user.

2. Using the Telescopic Slide Mounting Kit (35305D), which is available as an optional accessory. This method of rack-mounting, which also utilises the Rack Mounting Brackets provides full support for the logger, and also permits the logger to be easily pushed in and out of the rack. The Mounting Kit is suitable for racks having a depth of 30''.

Ventilation Adequate ventilation must be provided for ORION when it is fitted in a rack. This is particularly important if it is fitted above other instruments which generate appreciable heat. The peak power dissipation of ORION is 50W.

To fit ORION in a rack use either Method 1 or Method 2 below.

Method 1: Using the Rack Mounting Brackets and additional rear support.

1. Remove the finishers on both sides of the logger, see Fig 3.7. Each finisher is retained with 2 screws, keep these for fitting the Rack Mounting Brackets.

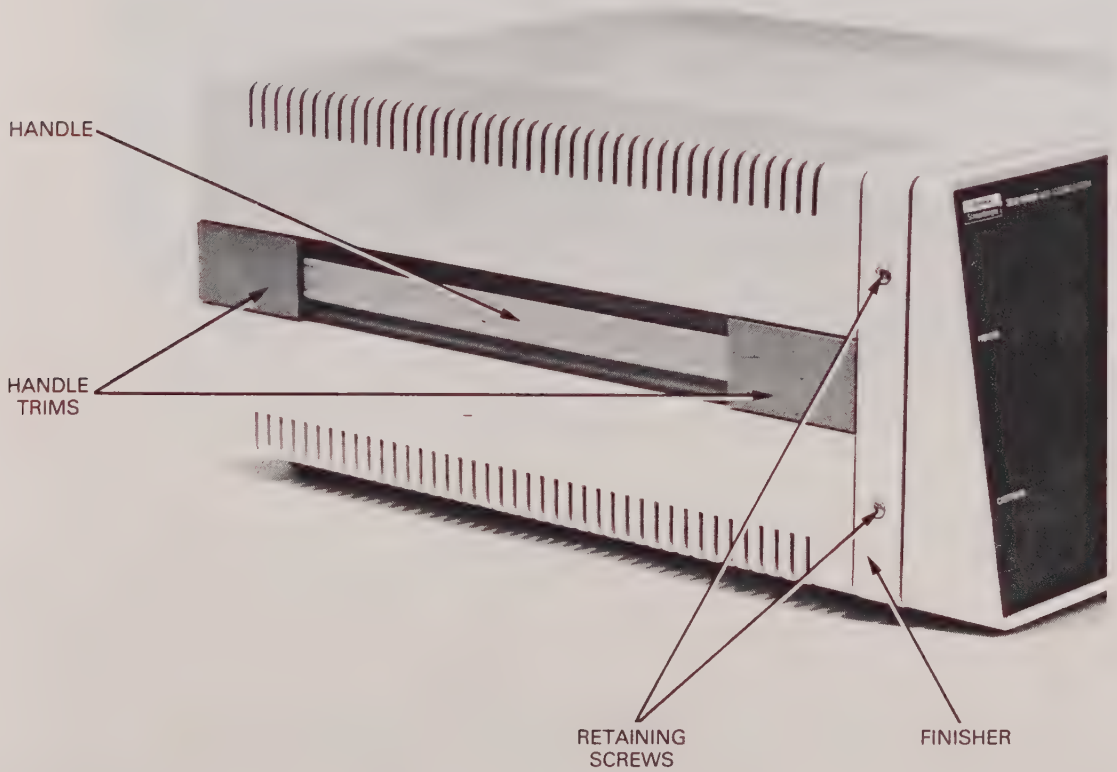


Fig 3.7 *Location of finishers and handle trims*

2. Remove all feet and tilt bar from underside of logger.

3. Fit the Rack Mounting Brackets in place of the finishers, using the retaining screws (see Fig 3.8). Note that the Brackets are normally fitted with the flange nearer the front of the logger (as shown) but they can be fitted the other way round (this would make the logger stand out farther in the rack).

4. Build an adequate support arrangement into the rack for supporting the rear of the logger. Note that the Rack Mounting Brackets are for retaining the logger only, they must not be used to support the entire weight of the logger.

5. Fit logger in rack. To secure, screw the Brackets to the rack.

Fig 3.8 *ORION with Rack Mounting Brackets fitted*

Method 2: Using Telescopic Slide Mounting Kit (35305D)

1. Remove the finishers on both sides of the logger, see Fig 3.7. Each finisher is retained with 2 screws, keep these for fitting the Rack Mounting Brackets.

2. Remove all feet and tilt bar from underside of logger.

3. Remove the handle trims from both sides of logger; use a penknife blade or similar tool and lever-out each trim on its upper or lower edge, see Fig 3.9. Remove both handles.

Fig 3.9 *Handle trim and finisher removed.*

4. Fit the Rack Mounting Brackets in place of the finishers, using the retaining screws (see Fig 3.8). Note that the Brackets are normally fitted with the flange nearer the front of the logger (as shown) but they can be fitted the other way round (this would make the logger stand our farther in the rack).

5. Take the rack bars from the Mounting Kit, and fit these to the logger in place of the handles; use the four countersink screws provided for each bar, see Fig 3.10.

6. Take the sliders from the Mounting Kit and remove the innermost slider (press spring to release). Fit one innermost slider to each rack bar using the 4 pan-head screws and washers provided for each slider; locate the slider such that it butts up against the rear edge of the Rack Mounting Bracket.

7. Fit the remaining parts of the telescopic slider assembly to the rack.

8. Pull out sliders from rack. Lift up the logger so that it is level with the sliders. Mate the sliders on the rack with the innermost sliders fitted on the logger. Ensure that the spring on each innermost slider locates in lug on moving slider.

9. Press the release springs on static part of slider assembly in rack and push logger into the rack. To secure, screw the Brackets to the rack.

Fig 3.10 *Fitting the Rack Mounting Kit 35305D*

4 SLAVE EXPANSION UNIT 3530E

The Slave Expansion Unit 3530E is designed specifically for use with the ORION Main Unit. Each Slave Unit permits an additional 200 analogue inputs to be connected. These inputs will be measured and processed by the ORION Main Unit in the normal way.

The Slave Unit has 10 slots at the rear (similar to the Main Unit) into which Reed Relay Selectors or Solid State Selectors may be fitted (see Fig 3.11). Note that analogue inputs only can be connected to a Slave Unit.

One, or two, Slave Units may be connected to a Main Unit. A logging system consisting of a Main Unit and two Slaves would provide 600 input channels. Interconnection between the Main Unit and Slave(s) is made using the cable(s) provided. Full details for installing a Slave Unit are given in the procedure below. It is important that these instructions are followed carefully, since correct operation of a Slave depends not only on proper interconnections but also on specific Modules being fitted in the Main Unit.

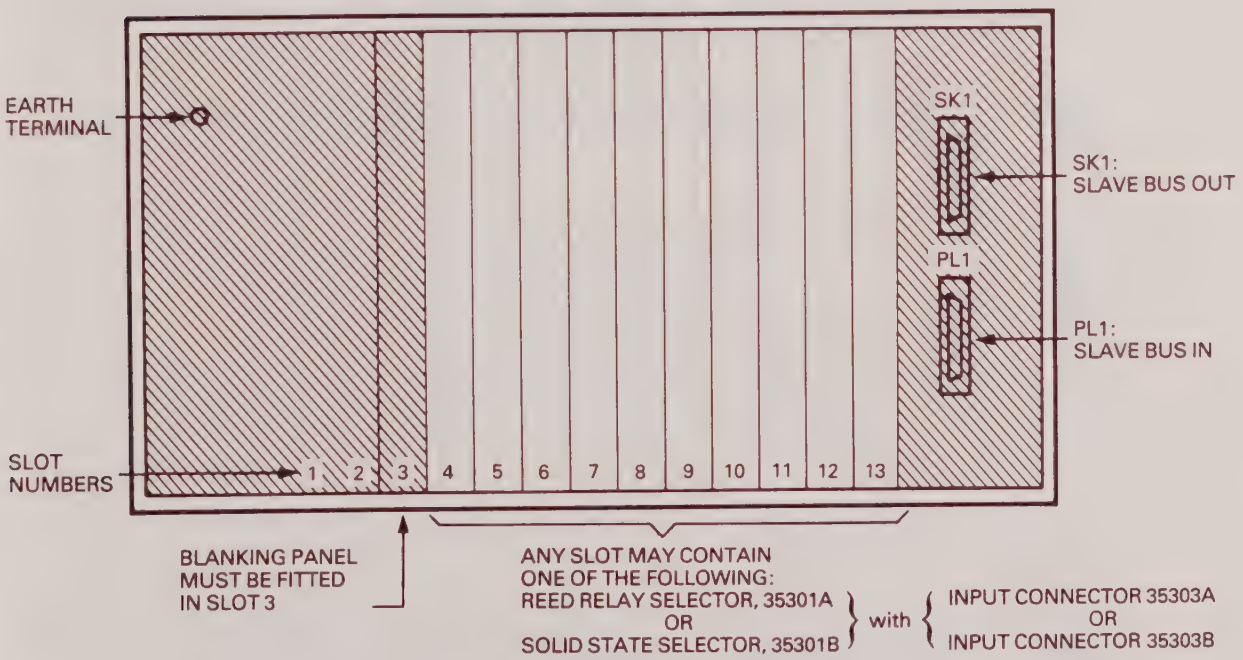


Fig 3.11 *Rear view of Slave Expansion Unit 3530E*

Equipment Required

Controller pcb 35300533
Transducer Energisation Module 35301D (may already be fitted in Main Unit)
Interconnecting cable 35300227A (1 per Slave required)

Installation Procedure

NB Slave Units can be used with ORION only if the modification status is B1 or higher (i.e. Controller pcb 53 and appropriate issue ROM's are fitted). If the status is lower than B1, consult a Solartron Service Department. The modification status is indicated on the Serial Number label which is located under the right hand carrying handle (as viewed from front of instrument).

1. Set power switch on ORION Main Unit to OFF.

2. If a Transducer Energisation Module (35301D) is already fitted to the Main Unit, remove the Module and connect the Analogue Bus Links (see diagram below), such that each solder pad is bridged. Refit the Module into slot 14 in the Main Unit. If the Module was not already fitted, take a new Transducer Energisation Module, connect the Analogue Bus Links, as previously described, and fit the Module into slot 14 in the Main Unit.

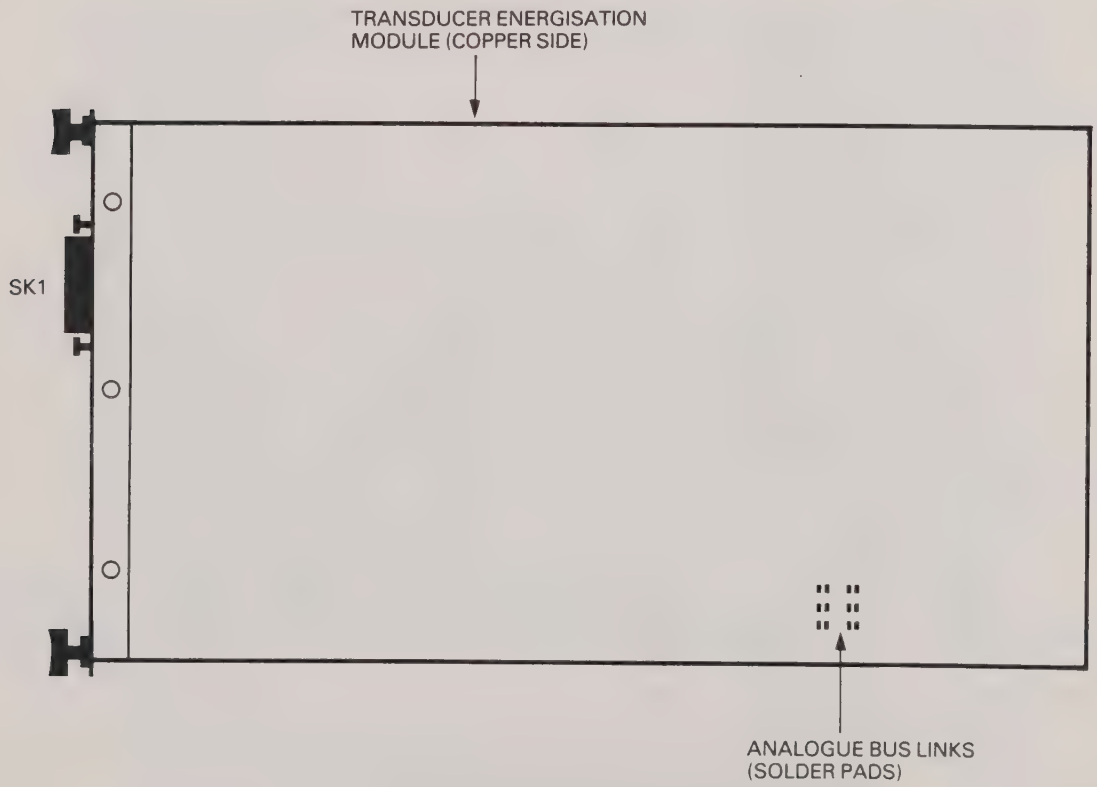


Fig 3.12 *Location of Analogue Bus Links on the Transducer Energisation Module*

3. Fit the Reed Relay Selectors (35301A) and/or Solid State Selectors (35301B) as required into slots 4 to 13 in the Slave Unit. Slot 3 must always have a blanking plate fitted.

   NB Analogue inputs only can be connected to the Slave Unit. Digital inputs cannot be connected, and the following Modules must **not** be fitted to a Slave Unit.
   Event-Status Input Module (35302A)
   Digital Input-Output Module (35302B)
   Analogue Output Module (35301E)
   Counter Timer (35302C)

4. Position the Main Unit and Slave(s) at the described location. If the Main Unit and Slave(s) are to be stacked, the preferred arrangement is for the Slave(s) to be stacked on top of the Main Unit, as shown in Fig 3.13.

5. Using the interconnecting cable(s) supplied (35300227A), connect the Main Unit to the Slave(s) as described in 5.1 and 5.2 below, and as shown in Fig 3.13.

   5.1 Connect SK1 on the Transducer Energisation Module in the Main Unit, to PL1 (Slave Bus In) on the Slave Unit.

   5.2 If using two Slaves, connect SK1 (Slave Bus Out) on the first Slave, to PL1 (Slave Bus In) on the second Slave.

6. Connect the measurement inputs to the Main Unit and the Slave Unit(s).

   Note: The Main Unit contains channels 001 to 200,
   the first Slave Unit contains channels 201 to 400,
   the second Slave Unit (if fitted) contains channels 401-600.

CAUTION The power switch on the Main Unit must be set to OFF before disconnecting or re-connecting the interconnection cables, or before any Module is removed or replaced. Before the Controller pcb is removed or replaced, the keyswitch must also be set to OFF.

7. Set the power switch on the Main Unit to ON. The system is now ready for use.

Notes

1. If two Slaves are connected to a Main Unit, at least one Reed Relay Selector or Solid State Selector must be fitted to either the Main Unit or to the 'first' Slave. This Selector must have an Input Connector fitted, even if no inputs are connected.

2. When a Slave is connected, the Specification for the ORION system is de-rated as detailed in Chapter 13 'Slave Expansion Unit 3530E'.

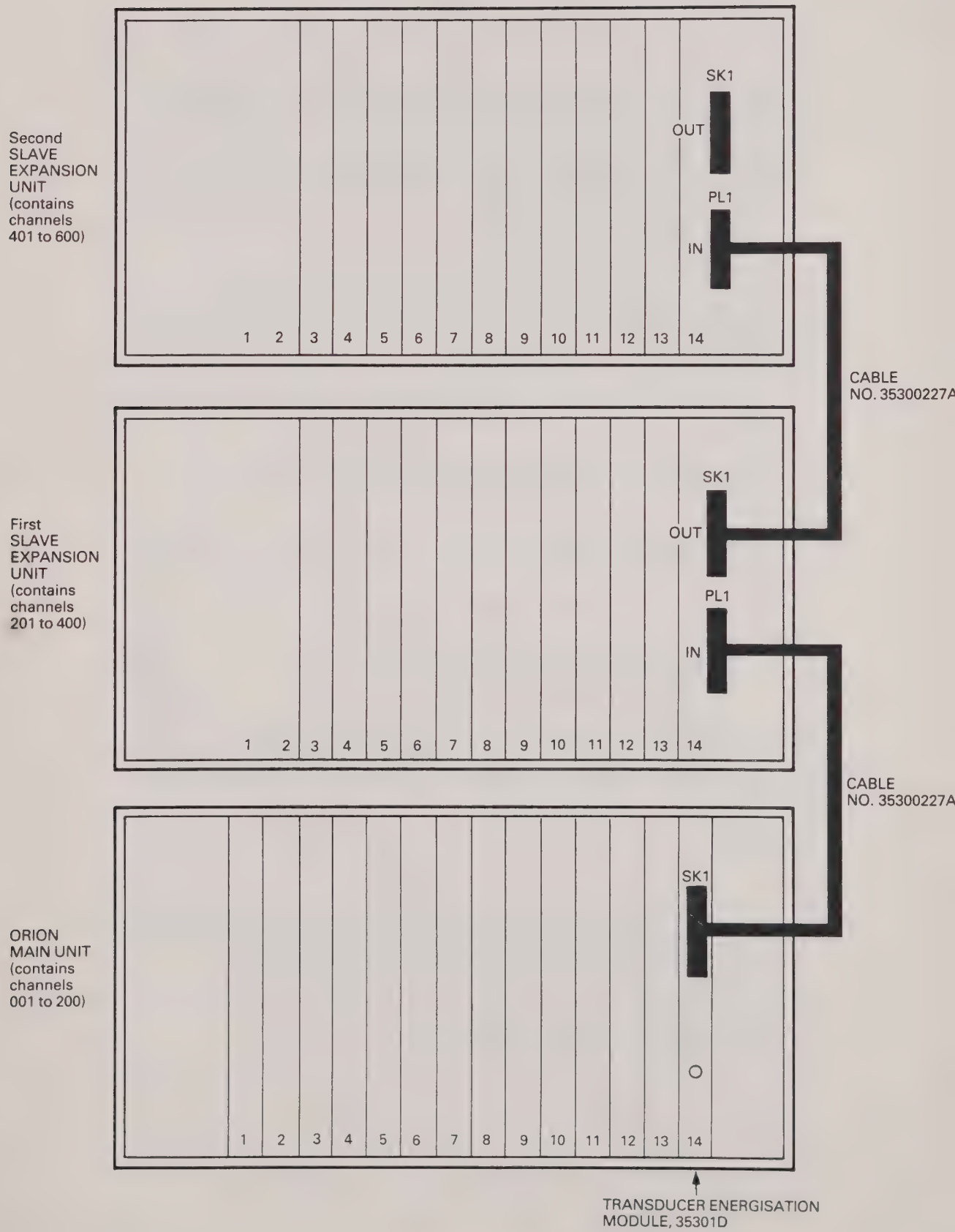

Second SLAVE EXPANSION UNIT (contains channels 401 to 600)
SK1
OUT
PL1
IN
1 2 3 4 5 6 7 8 9 10 11 12 13 14
CABLE NO. 35300227A
First SLAVE EXPANSION UNIT (contains channels 201 to 400)
SK1
OUT
PL1
IN
1 2 3 4 5 6 7 8 9 10 11 12 13 14
CABLE NO. 35300227A
ORION MAIN UNIT (contains channels 001 to 200)
SK1
O
1 2 3 4 5 6 7 8 9 10 11 12 13 14
TRANSDUCER ENERGISATION MODULE, 35301D

Fig 3.13 *Stacking arrangement and interconnections for the Main Unit and Slave Units*

Chapter 4
Using the Reed Relay or Solid State Selectors (35301A/B)

Contents

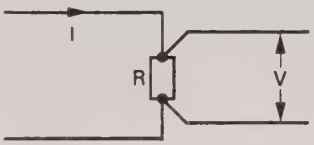
		Page No.
1	General Information	4.2
1.1	Current and Resistance Measurement	4.2
1.2	Energising currents	4.2
1.3	Strain Measurement	4.3
1.4	Common Mode and Series Mode Interference	4.5
1.5	Sense Amplifiers	4.7
1.6	Permitted inputs and sensors	4.7
1.7	Input Connectors	4.8
1.8	Channel Numbering	4.9
1.9	Selector Modules	4.12
1.10	Condensed Specification for Reed Relay and Solid State Selectors	4.12
2	Input Connections	4.13
2.1	General Procedure	4.13
2.2	Voltage Measurement	4.16
2.3	Thermocouple Measurement	4.18
2.4	2-terminal Resistance Measurement	4.22
2.5	4-terminal Resistance and PRT Measurement	4.24
2.6	Full Bridge Strain Gauge Measurement	4.26
2.7	¼-Bridge Strain Gauge Measurement	4.28
2.8	½-Bridge Strain Gauge Measurement	4.32
2.9	3-wire Strain Gauge Measurement	4.34
2.10	3-pole Current Measurement	4.38

1 GENERAL INFORMATION

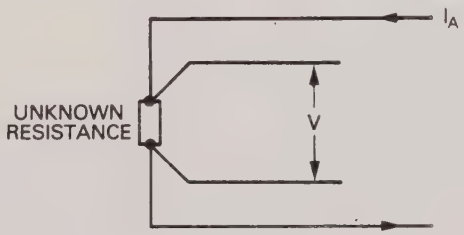
1.1 CURRENT AND RESISTANCE MEASUREMENT

The ADC (analogue-to-digital converter) in the logger will directly measure dc and ac voltages. In order to measure other quantities, such as resistance and temperature, a signal which represents the unknown quantity is fed to the ADC as a voltage.

In order to measure the value of an unknown current (I), the current is passed through a resistor (R) of known value; the resultant voltage (V) developed across the resistor is a measure of the unknown current, where $I = \frac{V}{R}$



In a similar manner, the value of an unknown resistance can be determined by passing a known value of current through the resistance and measuring the voltage developed across it. In practice the known energising current I_A, derived from the logger, is used for this purpose.



The above configuration shows '4-terminal' resistance measurement, this arrangement is used for some inputs to the logger. As the diagram shows, current is passed through one pair of wires while the other pair of wires is used to measure the voltage developed across the resistance only. The voltage connections are made as close to the resistor as possible, hence minimising the effect of any voltage drop in the current-carrying leads. The values of voltage measured are automatically converted to appropriate units of current or resistance within the logger.

1.2 ENERGISING CURRENTS

The logger provides two current sources, I_A and I_B, which are used for sensor energisation. Both sources can be set to 1 of 4 current levels; either source can be switched on and off independently.

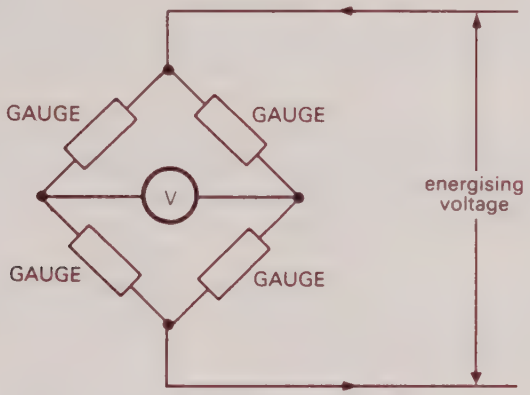
The sources are used in different ways depending on the input/sensor type being measured. The configuration of current sources and signal input lines is controlled automatically using the internal relays. (Voltage energisation is used for full bridge strain gauge inputs).
The procedure for setting I_A and I_B is given on page 4.15.

1.3 STRAIN MEASUREMENT

The logger will measure strain gauges connected in any of the following configurations.

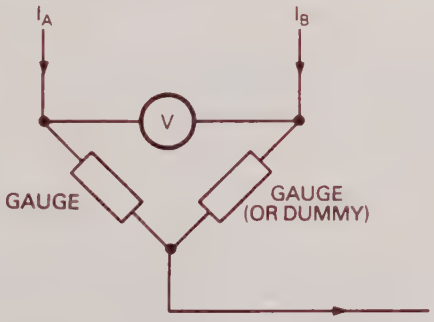
Full bridge
½ bridge
3-wire
¼ bridge, with local or remote common dummy.

Full bridge



A full bridge configuration can consist of 4 separate gauges or one composite gauge. Alternatively, one or more of the components can be an unstrained dummy gauge used for temperature compensation.

½ bridge

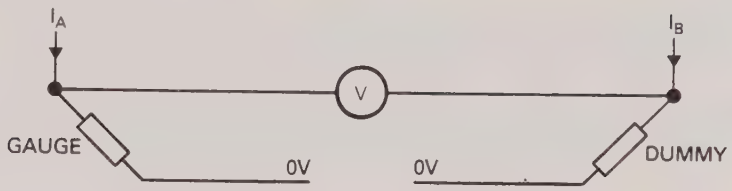


A ½ bridge configuration can consist of 2 gauges, or 1 gauge and a dummy gauge. When both components are gauges, they could be bonded to the test structure such that one is in compression while the other is in tension, thereby increasing measurement sensitivity. ½ bridge configurations are energised by two current sources and are measured using the Solartron Dual Current technique which gives linear measurements.

3-wire

Strain gauge installations that use the '3-wire' connection can be used with the ORION logger. The benefits of dual-current energisation are retained and the wiring is simpler than that need for full bridges. The electrical arrangement is similar to that for ½ bridge except that the dummy is located at the logger (it is connected directly across the input terminals). For further details see page 4.34.

¼ bridge with local or remote dummy



The ¼ bridge with local or remote dummy provides the simplest and most economic method of strain measurement. Although called '¼ bridge', the gauge and dummy together actually form a ½ bridge circuit which, like the true ½ bridge configuration, is measured using the Solartron Dual Current technique.

The dummy should have an identical nominal resistance and temperature coefficient to that of the gauge, it can be located close to the gauge itself (local dummy) or at some other location away from the gauge (remote dummy).

Gauge Factor

Gauge Factor is the ratio of relative change in resistance to the relative change in length due to strain. Gauge Factor (G) is given by the following formula: $G = \frac{\Delta R/R}{\Delta L/L}$

where, R = gauge resistance (unstrained)
ΔR = change in resistance due to strain
L = gauge length (unstrained)
ΔL = change in length due to strain

Typically, the gauge factor for a conventional wire strain gauge is in the range 2 to 2.2.

When entering the gauge factor, as part of the operating program, it is essential to enter the effective total gauge factor of the configuration. For ¼ bridge configurations, this is simply the gauge factor of the single gauge in the ¼ bridge. For ½ and full bridge configurations the effective gauge factor is generally given by summing the gauge factors of each active (and re-inforcing) gauge.

e.g.1. for a full bridge consisting of 4 active gauges, add together the gauge factor of all 4 gauges; assuming each gauge has a factor of 2, total = 8.

e.g.2 for a ½ bridge consisting of 2 active gauges, add together the gauge factor of each gauge; assuming each gauge has a factor of 2, total = 4.

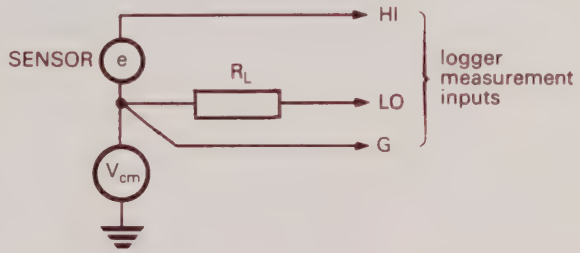
Include only active gauges in the calculation; omit non-active gauges (fitted for temperature compensation, etc). Thus, for a full bridge with 2 active gauges, and 2 temperature-compensating gauges, add the factors of the two active gauges only. All gauges included should be physically mounted such that their outputs reinforce, i.e. one in tension, the other in compression.

1.4 COMMON MODE AND SERIES MODE INTERFERENCE

Measurement errors can occur if common mode or series mode interference is present. Precautions can be taken by the user, however, to keep such errors to a minimum.

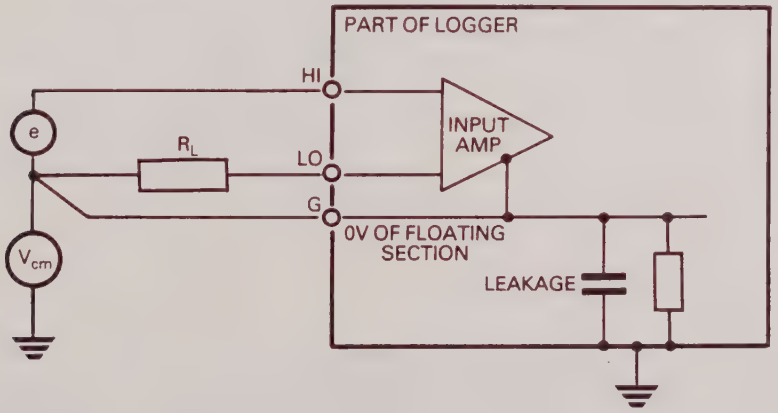
Common Mode Interference

A typical input circuit is shown in the simplified theoretical diagram below.



The voltage from the sensor to be measured is represented by e. It is possible that the sensor will have some standing voltage with respect to ground; this is the common mode (interference) voltage, represented by V_{cm}. The resistances of the Hi and Lo connecting leads are likely to be unequal, this is represented by R_L in the Lo lead. The input circuit is connected to the logger measuring circuit using the Hi, Lo and G (guard) input terminals.

The diagram below shows an input circuit connected to the logger. Note that G is connected directly to the source of common mode voltage V_{cm}; this arrangement provides the best immunity to common mode interference.



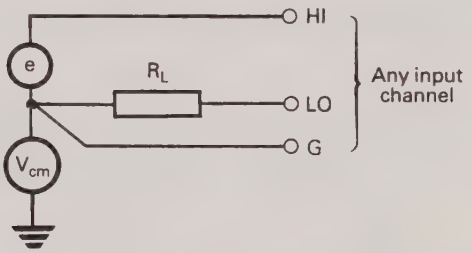
Internal Leakage

Both Hi and Lo inputs in the logger have high impedance, and the leakage between these and earth is insignificant. Even with the best engineering however, some leakage will exist between the 0V floating section of the logger and mains earth. The common mode voltage V_{cm} will cause current to flow via the G connection and through the leakage path back to mains earth. This will not degrade measurement accuracy providing that the G terminal is connected to V_{cm} as shown above. The leakage will however cause some measurement error if the G terminal is connected to the Lo terminal, as may be necessary in some applications.

Mandatory connection of Guard

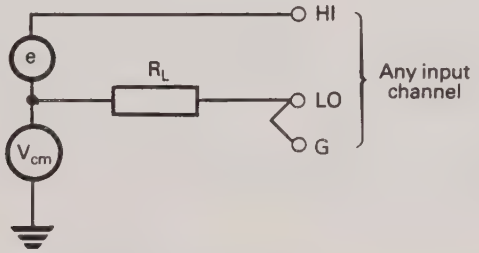
In order that the measurement circuit can operate correctly it is mandatory for the G (guard) connection to be made, as the 0V floating section in the logger must float to the potential of any common mode interference voltage. There are two ways to connect Guard:

(a) Directly to the source of common mode voltage.



This is the preferred method which will provide the best rejection of common mode interference.

(b) To the Lo terminal.



This method should be used where method (a) above is not practicable. It gives less rejection of common mode interference as V_{cm} will cause current to flow through R_L and the internal leakage path to earth. The current flowing through R_L will produce a voltage drop and thereby cause a certain amount of measurement error. It is important, nevertheless, to connect Guard as shown in (a) or (b) above, as gross errors may occur if no Guard connection is made, leading to possible measurement overload.

Series Mode Interference

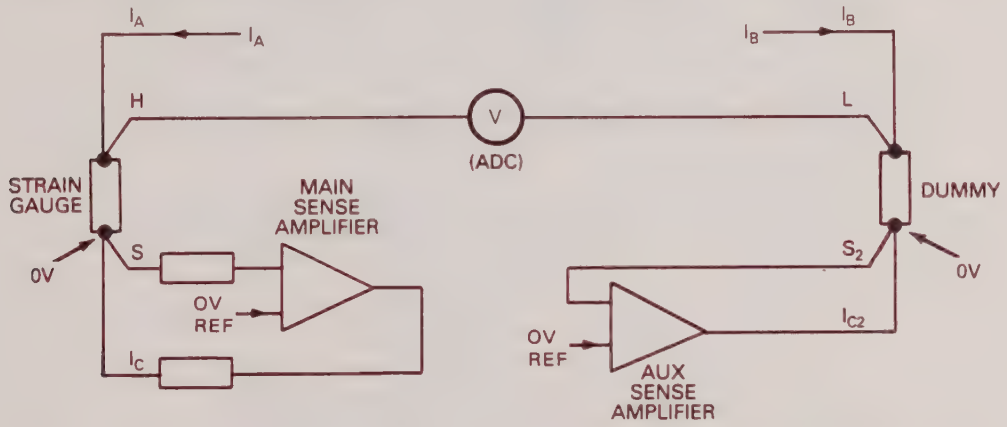
Series mode interference cannot be alleviated in the same way as common mode interference. No measuring instrument can provide immunity to dc series mode interference but the logger does have built-in rejection to ac series mode interference. This is particularly useful since one of the most usual types of series mode interference is that picked up from the ac mains supply.

Such rejection of ac mains interference is given when the logger is set to one of the two slower measurement rates (40/s and 10/s). At the 40/s rate the adc provides an average of the input over the 20ms integration time, thereby virtually eliminating superimposed interference at 50Hz and multiples thereof. At the 10/s rate, the adc has an 80ms integration time which gives similar rejection of ac at 12.5Hz, 25Hz, 50Hz, etc.

Rejection of mains frequency interference is not provided when either of the two faster measurement rates (500 or 150/s, 100/s) are used. In electrically noisy environments, where mains interference may be picked up, it is therefore essential to make measurements at either 40/s or 10/s.

1.5 SENSE AMPLIFIERS

The logger contains two differential Sense amplifiers which are automatically configured by the logger according to the type of input or sensor defined for each input. A simplified diagram showing the Sense amplifiers as configured for strain measurement (using ¼ bridge with remote dummy) is shown below.



The main sense amplifier monitors the S(sense) connection at the strain gauge and controls the output (I_c) such that a potential of 0V is maintained at the gauge. The auxiliary sense amplifier maintains the S_2 connection to the dummy at 0V, in a similar manner. The sense amplifiers thereby maintain both points at 0V without the need to connect the points together or to use floating current supplies. This effectively compensates for any voltage drop which may be present in the current-return lines I_c and I_{c2}.

1.6 PERMITTED INPUTS AND SENSORS

The following types of input and sensors may be measured directly:

Voltage dc (1μV to 110V)*	*applies to Reed Relay Selector: (the maximum input to Solid State Scanner is 14V).
Voltage ac (10μV to 20V)*	
Resistance, including PRT's and thermistors	(1mΩ to 130K, using 100μA energising current).
Strain gauge (120Ω or appropriate resistance)	
Full bridge	
¼ bridge with common dummy (local or remote)	
½ bridge	
Thermocouple (types E, K, J, R, S, T)	
Current dc or ac (0.1μA to 25mA)	using 35303B Input Connector, or 35303A Input Connector with additional 100Ω shunt resistors.

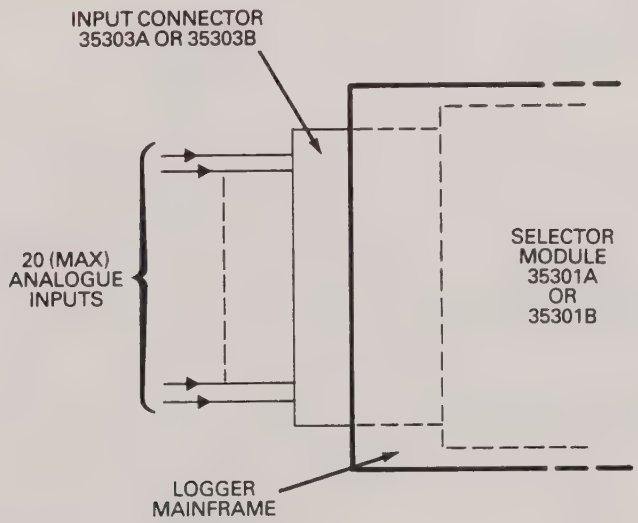
1.7 INPUT CONNECTORS

Inputs and sensors are connected to the logger using screw terminals in the Input Connectors. Two types of Input Connector are offered, 35303A and 35303B.

35303A Can be used for all types of input. 2 thermistors are built in; these are used for sensing Ambient Temperature in thermocouple measurements (cold junction) and for use in drift-correct measurements.
For current inputs: Fit an additional 100Ω shunt resistor across each current channel, between H and L terminals (shunt resistors having other values of resistance can be used). Select voltage measurement for the channel(s) (i.e. Select Measurement Type, VOLTS) and use a CONVERSION to convert the voltage into current.

35303B Intended primarily for current measurements; a 100Ω shunt resistor is fitted internally across each input channel.
Can also be used for any other types of input providing the input impedance is high. (Thermistors are not fitted).

Each Input Connector mates with a Reed Relay or Solid State Selector in the logger mainframe. The two types of Input Connector are physically interchangeable.



1.8 CHANNEL NUMBERING

Fig 4.1 shows an Input Connector assembly with the cover removed. The simplified diagram shows the rows of screw terminals to which the inputs may be connected.

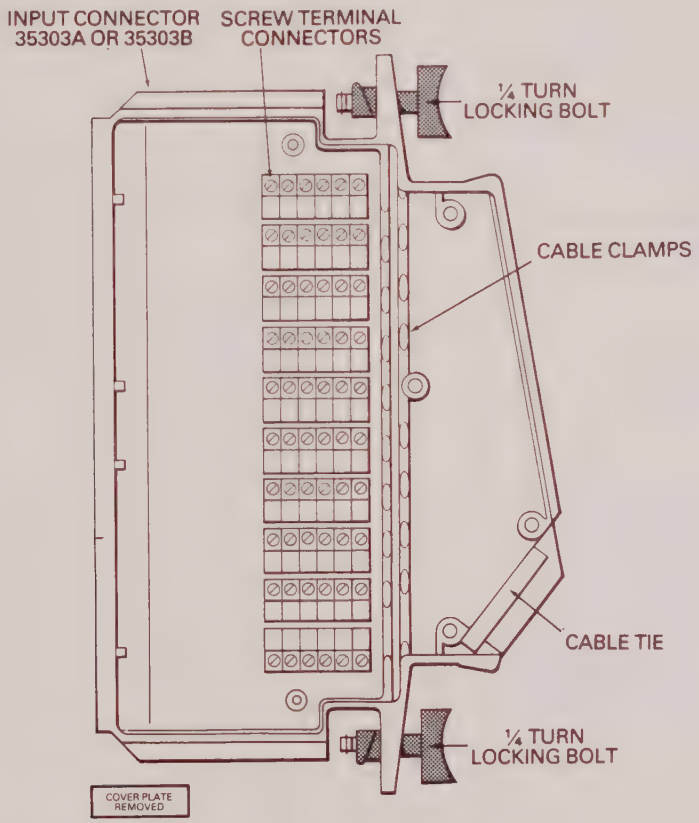
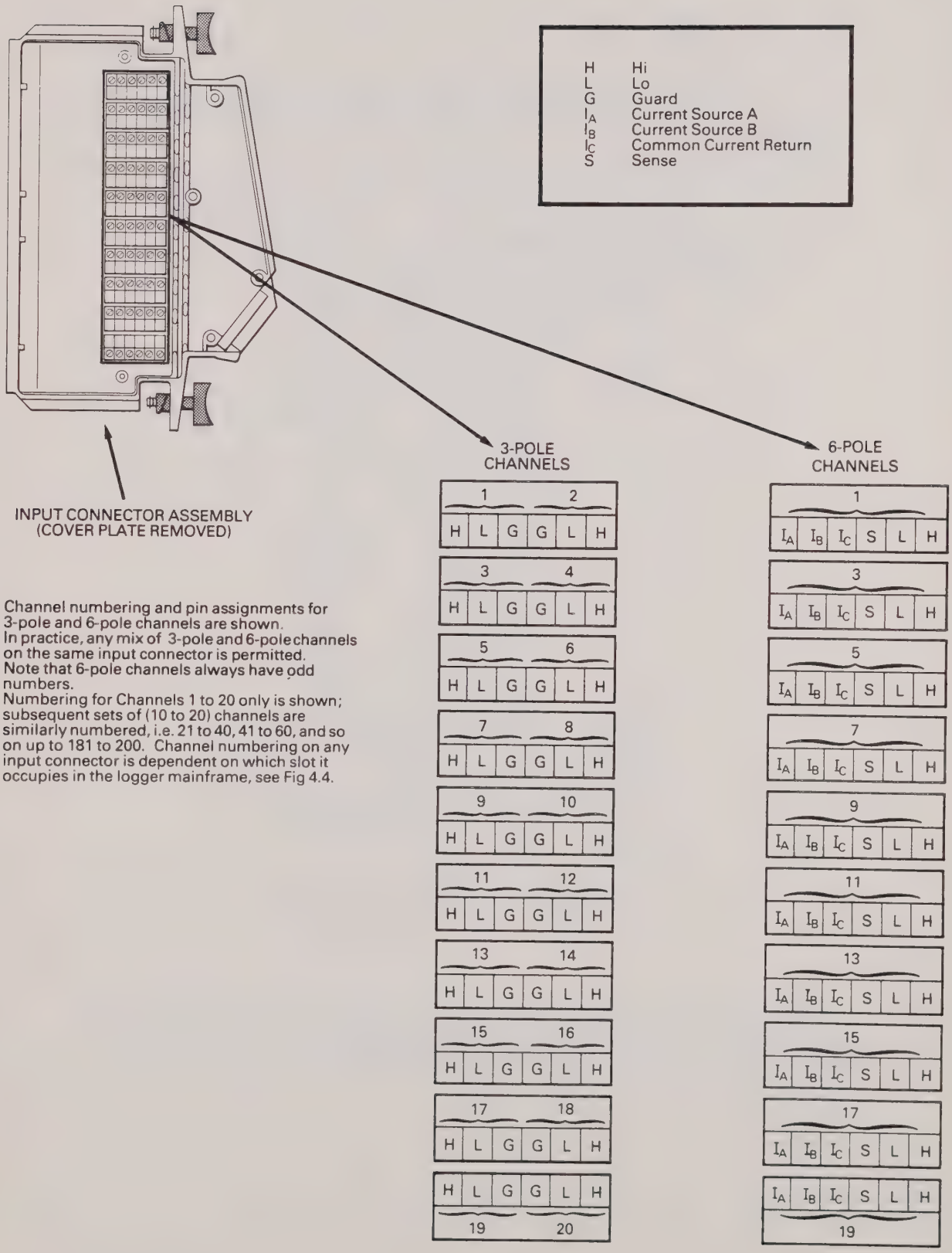


Fig 4.1 *Input connector with cover removed*

The input channels are numbered in groups of 20 i.e. 1-20, 21-40, etc. Fig 4.2 shows channel numbering and assignment of input terminals. Fig 4.3 shows channel numbering and pin assignment for a typical mixture of 3-pole and 6-pole input channels. Fig 4.4 shows the channel number groups, these being dependent on the location of the Input Connector in the mainframe.



Channel numbering and pin assignments for 3-pole and 6-pole channels are shown.
In practice, any mix of 3-pole and 6-pole channels on the same input connector is permitted.
Note that 6-pole channels always have odd numbers.
Numbering for Channels 1 to 20 only is shown; subsequent sets of (10 to 20) channels are similarly numbered, i.e. 21 to 40, 41 to 60, and so on up to 181 to 200. Channel numbering on any input connector is dependent on which slot it occupies in the logger mainframe, see Fig 4.4.

Fig 4.2 *Channel Numbering and Assignment of Terminals on the Input Connectors*

Pin assignments are automatically set by the software for 3-pole or 6-pole channels.
i.e. H L G for 3-pole
I_A I_B I_C S L H for 6-pole
Note that 6-pole channels always have odd numbers, the next corresponding even number is omitted.

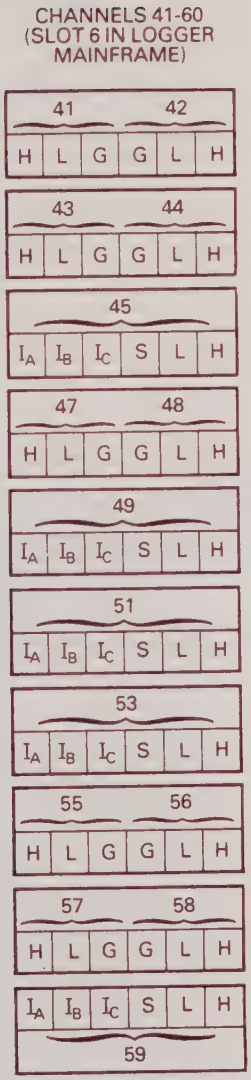
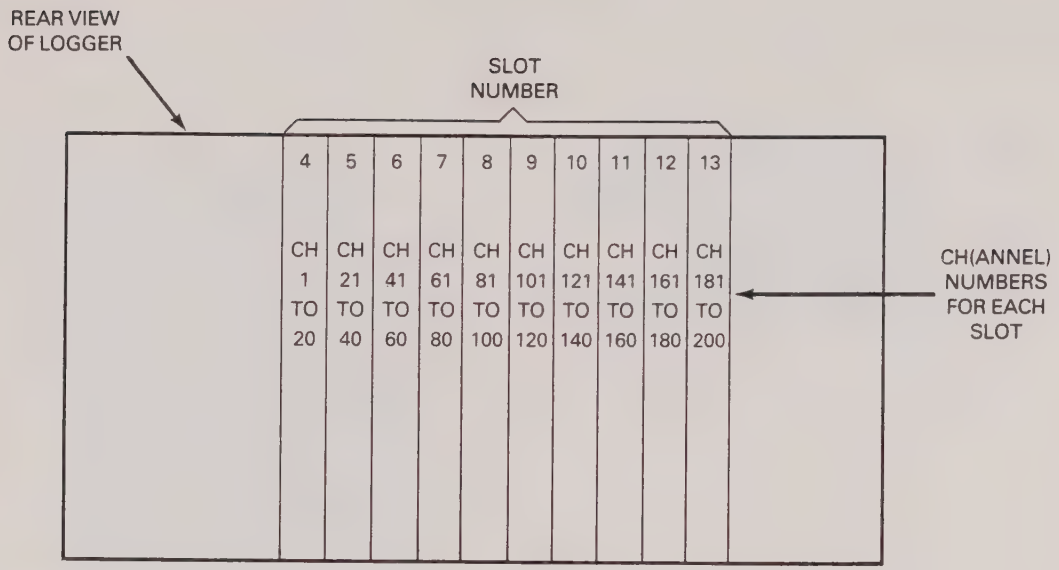


Fig 4.3 *Channel Numbering and Pin Assignment for a typical mixture of 3- and 6-pole channels*



An Input Connector and the associated Reed Relay or Solid State Selector may be fitted in any of the slots 4-13 in the logger mainframe. The Connectors and the Selectors are interchangeable; channel numbering is however always dependent on position in the mainframe.

Fig 4.4 *Channel Numbering in logger mainframe*

1.9 SELECTOR MODULES

Two types of Selector are offered for analogue inputs:

Reed Relay Selector	35301A
Solid State Selector	35301B

One Selector of either type can occupy any of the mainframe slots 4-13 inclusive; any one logger can thus be fitted with up to 10 Selectors.

Main advantages of each type The Reed Relay type has the ability to handle high input and common mode voltages, whereas the Solid State type has faster switching (up to 500/s) and solid-state reliability.

N.B. Should any logger be fitted with a mixture of Reed Relay and Solid State Selectors, the input and common mode voltage must be restricted to that specified for Solid State Selectors. This applies even if the Solid State channels are not in use. It is therefore recommended that Selector types are not mixed in any one logger frame.

A condensed specification for each type of Selector is given below, this will facilitate selection of the correct type for any particular application.

1.10 CONDENSED SPECIFICATION FOR REED RELAY & SOLID STATE SELECTORS

	REED RELAY SELECTOR	SOLID STATE SELECTOR
Maximum input voltage (for full scale reading)	110V	14V
Maximum common mode voltage	200V	15V (any line to earth)
Number of channels and poles	10 × 6 pole or 20 × 3 pole or any mixture of 3- and 6-pole channels	10 × 6 pole or 20 × 3 pole or any mixture of 3- and 6-pole channels
Maximum scanning speed (channels/second)	150 ch/s	500 ch/s

2 INPUT CONNECTIONS

2.1 GENERAL PROCEDURE

1. Refer to the preceding Introduction to determine the correct type of Input Connector and Selector Module for the particular inputs to be connected.

2. Remove the Input Connector from the logger mainframe by turning the two ¼-turn locking bolts anti-clockwise and withdrawing the assembly, see Fig 4.5.

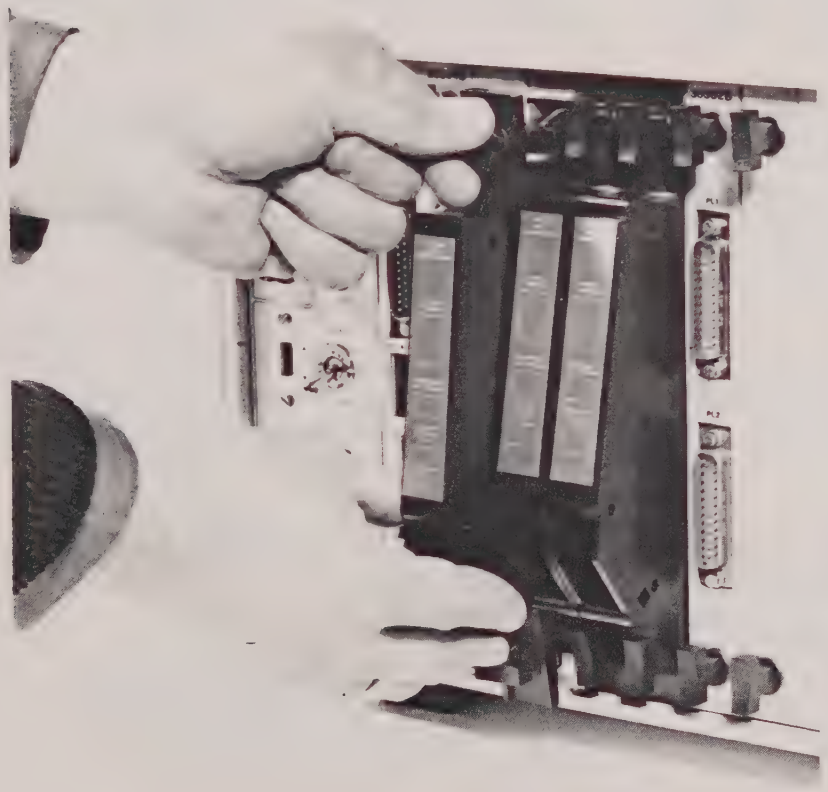


Fig 4.5 *Removing an Input Connector*

3. Remove the cover plate from the Input Connector to reveal the screw terminal blocks.

4. See section 1.8 'Channel Numbering' to determine channel numbering and assignment of terminals.

5. Connect the inputs/sensors to the input terminals, as shown in the relevant connection diagrams, see list below.

NB. GUARD (or SENSE) must always be connected:
For all 3-terminal inputs, GUARD must be connected to either the source of common mode, or directly to the LO input. For all 6-terminal inputs, SENSE must be connected as indicated in the appropriate connection diagrams.

INPUT/SENSOR TYPE	*Page No.*
Voltage	4.16
Thermocouple	4.18
Resistance or PRT	4.22
Strain, full bridge	4.26
Strain, ¼ bridge	4.28
Strain, ½ bridge	4.32
Current	4.34

There are 3 diagrams for each type of input/sensor:

(a) Theoretical Diagram; shows the measurement principle involved.
(b) Practical Connection Diagram; facilitates connection of the input/sensor to the screw terminals.
(c) Internal Signal Organisation; provided for general information, shows the input interconnection in the logger including current sources, sense amplifiers and the ADC.

6. When all the connections to the Input Connector have been made, loop the leads neatly around the cable clamps and through the cable tie, see Fig 4.1. Do not pass connecting leads over the top of the terminal blocks as this will prevent the cover plate from fitting properly, and will cause the leads to be trapped between the sharp edges on the blocks and the cover.

7. If desired, enter any input information on the 'write-on' patch on the cover plate label. Replace the cover plate on Input Connector.

8. Replace Input Connector.

9. If any inputs use the energising currents I_A or I_B, and it is necessary to select the user definable current value, see the information and instructions on page 4.15.

Setting the User-Defined Values of Energising Currents I_A and I_B

The two energising currents I_A and I_B are produced by the Transducer Energisation Module which is located in slot 14. I_A and I_B can be programmed to produce one of 4 levels of current; these are 8.33 mA, 1 mA, 100μA, or a user defined value.

The user-defined values for I_A and I_B are set independently by fitting 2 additional resistors to the pcb. The value of each resistor is determined by $R = \frac{6}{I}$ where, R = resistance (Ω) and I = current (amps).

The resistor for I_A is connected between P200 and P201. The resistor for I_B is connected across P202 and P203.

Normally, both I_A and I_B are set to identical values but can be different if required. The user defined current can be set to any value between 10mA and 100μA.

To fit resistors:

9.1 Switch off logger. Remove the Transducer Energisation Module from slot 14 in the mainframe.

9.2 Depending on the user-defined level of I_A and I_B, fit a suitable resistor across P200 and P201 for I_A, and another across P202 and P203 for I_B. See below for location of P200-203.

9.3 Replace Transducer Energisation Module in slot 14.

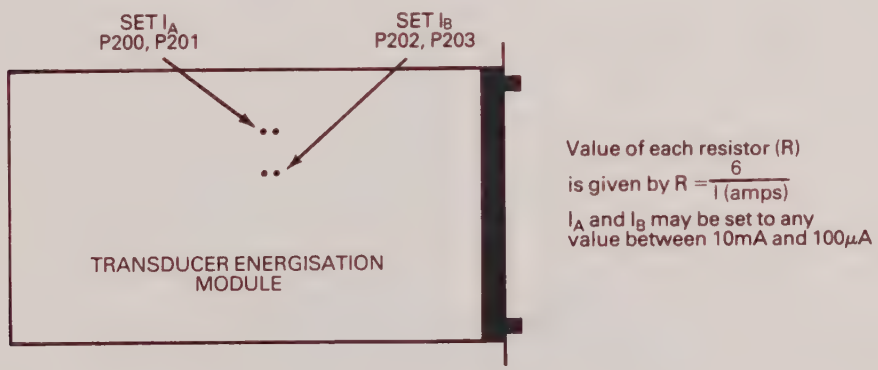
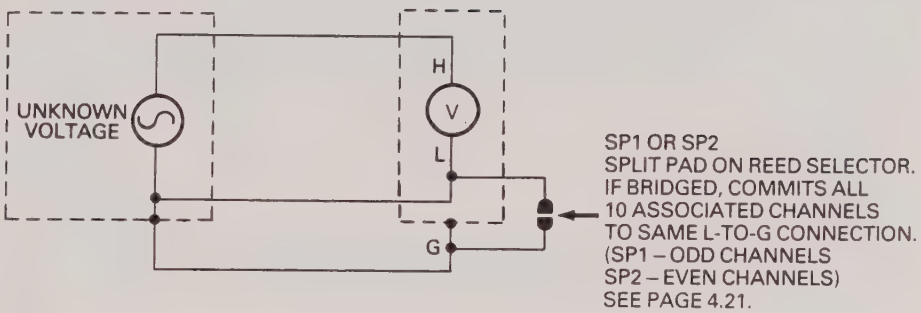

SET I_A
P200, P201
SET I_B
P202, P203
TRANSDUCER ENERGISATION MODULE
Value of each resistor (R)
is given by $R = \frac{6}{I\ (amps)}$
I_A and I_B may be set to any value between 10mA and 100μA

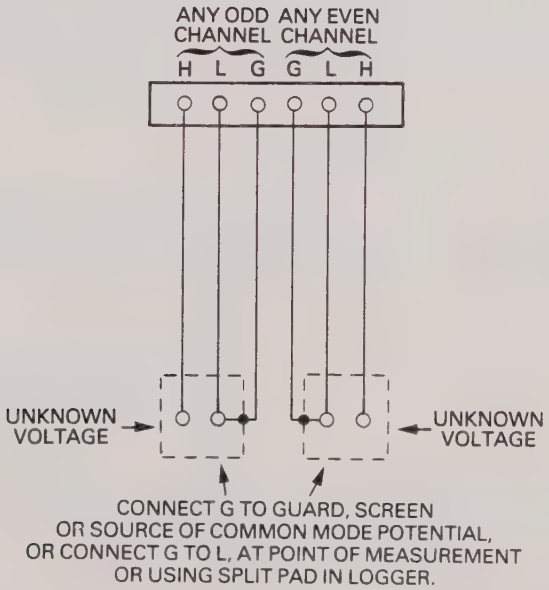
Fig 4.6 *Location of current-defining resistors on Transducer Energisation Module*

2.2 VOLTAGE MEASUREMENT

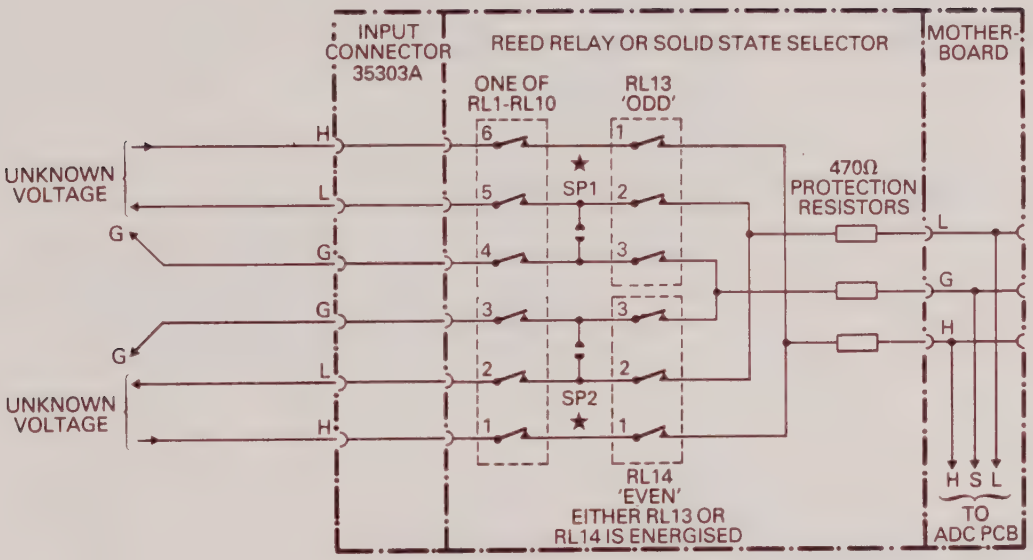
2.2.1 Theoretical Diagram



2.2.2 Practical Connection Diagram


ANY ODD CHANNEL
ANY EVEN CHANNEL
H L G G L H
UNKNOWN VOLTAGE
UNKNOWN VOLTAGE
CONNECT G TO GUARD, SCREEN
OR SOURCE OF COMMON MODE POTENTIAL,
OR CONNECT G TO L, AT POINT OF MEASUREMENT
OR USING SPLIT PAD IN LOGGER.

2.2.3 Internal Signal Organisation

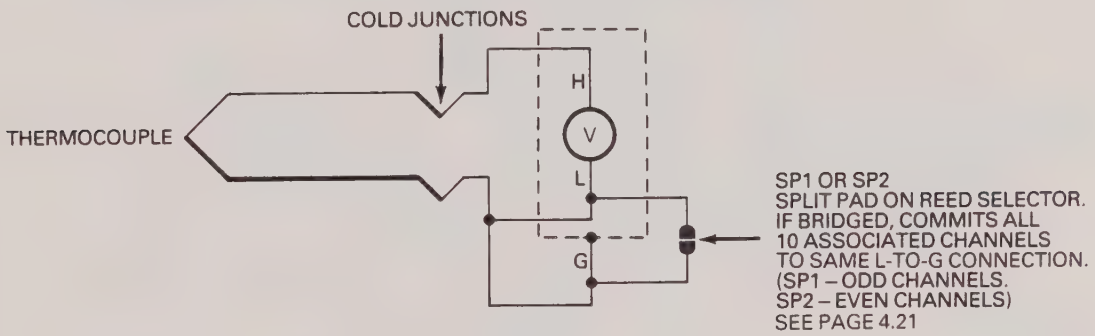


★ NOT ON SOLID STATE SELECTOR

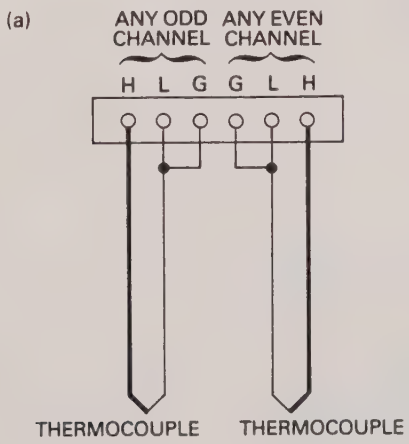
2.3 THERMOCOUPLE MEASUREMENT

See notes on Thermocouple Measurement, page 4.20.

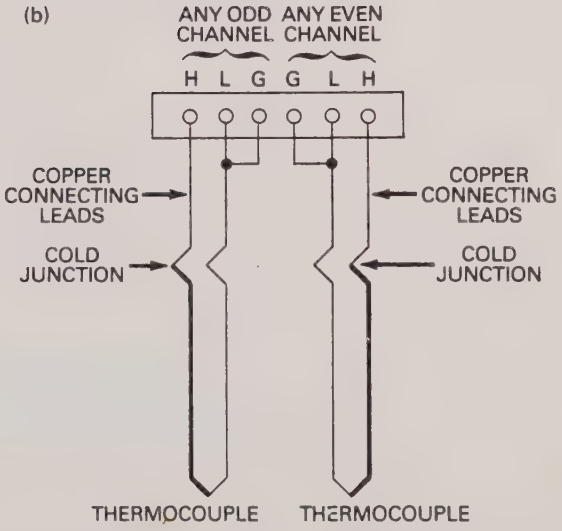
2.3.1 Theoretical Diagram



2.3.2 Practical Connection Diagrams

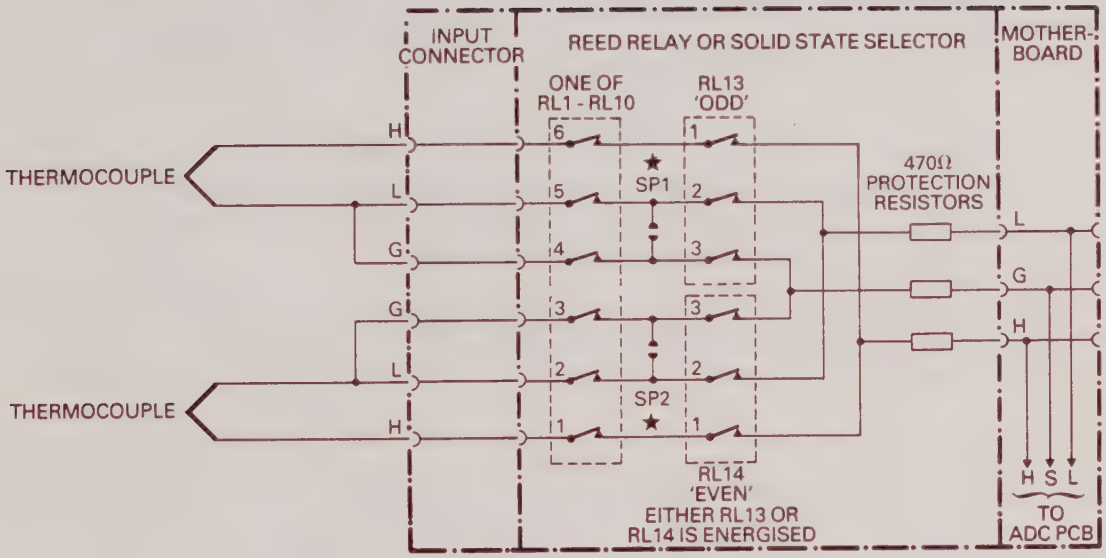


Method (a)
Thermocouple leads taken back to input terminals. Select AMB cold junction temperature compensation. Use Input Connector 35303A only. Cold junction (ambient temperature) is sensed in Input Connector.



Method (b)
Thermocouple leads connect to copper connecting leads, forming the 'cold junction' which is at a known temperature e.g. REF1-4 or 0°C. Select REF1-4 (and set temperature using INITIALISE, References) or select °0C.
Use either type of Input Connector 35303A/B.

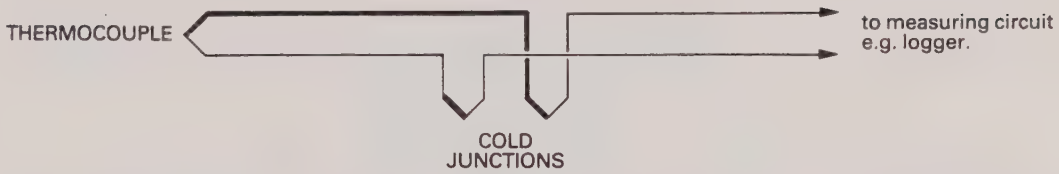
2.3.3 Internal Signal Organisation



★ NOT ON SOLID STATE SELECTOR

Thermocouple Measurement

The diagram below illustrates a typical thermocouple circuit.

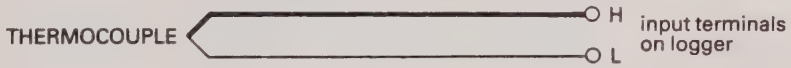


Note that a 'cold junction' (of dissimilar metals) is always formed where the thermocouple leads connect to copper connecting leads or input terminals. The cold junction temperature can, in practice, be at ambient or some other known reference.

Thermocouples may be connected to ORION using one of two methods, (a) and (b). Each method provides a different cold junction arrangement.

Method (a), see also 2.3.2 (a) on page 4.18.

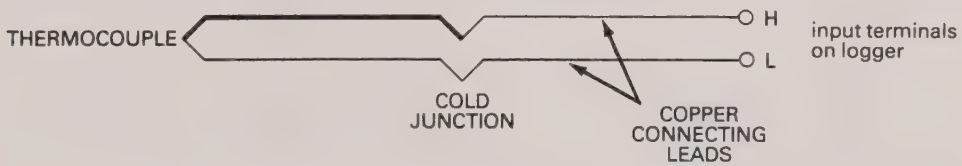
The thermocouple leads are taken directly to the input terminals.



In this case the input terminals are the 'cold junction', the temperature of which is sensed by thermistors built into the Input Connector (35303A only). When using this method, select 'Ambient' Cold Junction Compensation when setting up the logger; this accesses the sensing thermistors. Use Input Connector type 35303A only (the 35303B type does not have built-in thermistors for detection of 'ambient' temperature).

Method (b), see also 2.3.2 (b) on page 4.18.

The thermocouple leads are connected to copper connecting leads, forming the cold junction, and the copper leads are taken to the input terminals.



This arrangement permits the cold junction to be placed at any location, as required by the user, at which the precise temperature is known. This temperature could for instance be 0°C (e.g. melting ice) or any other known reference (Ref 1-4). When using this method, select one of the following for 'Cold Junction Compensation':

Reference 1,
Reference 2,
Reference 3,
Reference 4,
or 0°C.

The temperature for References 1-4 is set using 'INITIALISE References'.

Either type of Input Connector (35303A or B) can be used for method (b).

G (Guard) linking using SP1 and SP2 on Reed Relay Selector.

If the Reed Relay Selector is used and all odd or even input channels are to be used for similar voltage or thermocouple inputs then the external G (guard) to L (Lo) connection can be omitted providing that either SP1 or SP2 on the Reed Selector is linked. SP1 links G to L on all odd channels, SP2 links G to L on all even channels.

If either link is made the input connections are simplified, but it should be noted that all the associated channels are committed to the same G to L connection and that the best immunity to common mode voltage may not then be provided.

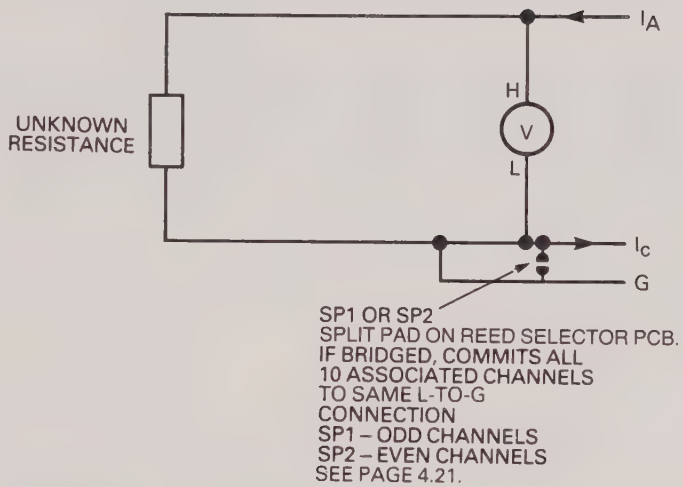
To link SP1 or SP2

1. Switch off logger.
2. Remove Reed Relay Selector.
3. Bridge SP1 and/or SP2 with solder.
4. Replace Reed Relay Selector.

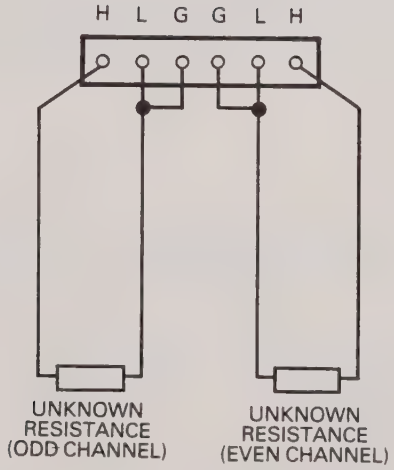
Fig 4.7 *Location of SP1 and SP2 on Reed Relay Selector*

2.4 2-TERMINAL RESISTANCE MEASUREMENT

2.4.1 Theoretical Diagram

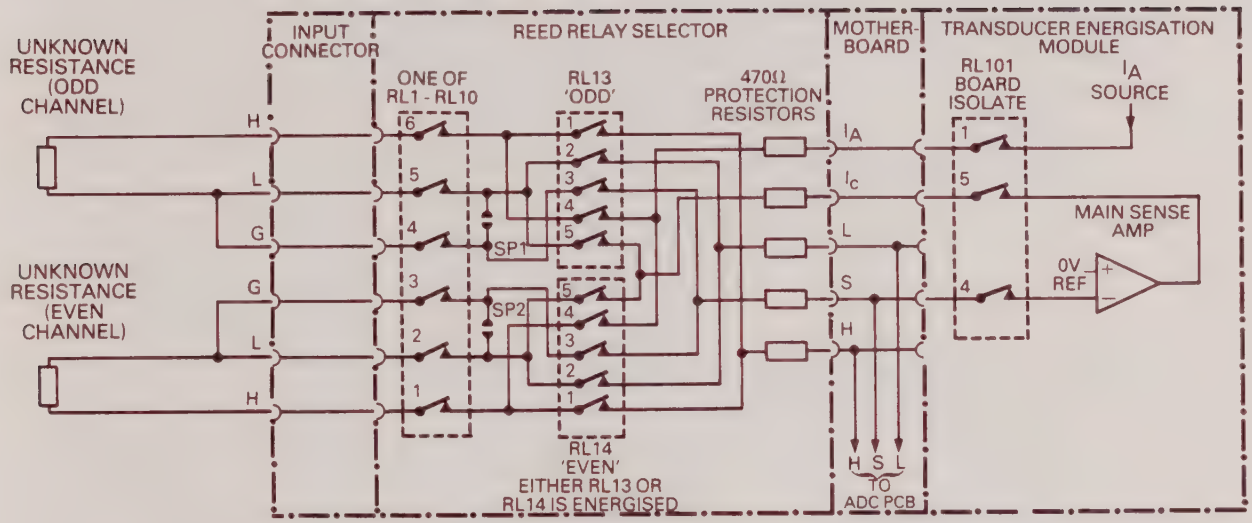


2.4.2 Practical Connection Diagram



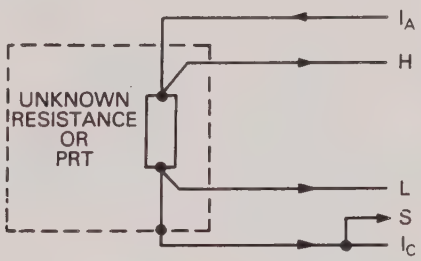
N.B. Use Reed Relay Selector 35301A only. 2-terminal resistance measurement is suitable for applications where less accuracy is required (e.g. measuring 10k to 1% accuracy). For greater accuracy, use 4-terminal resistance measurement, see page 4.24.

2.4.3. Internal Signal Organisation



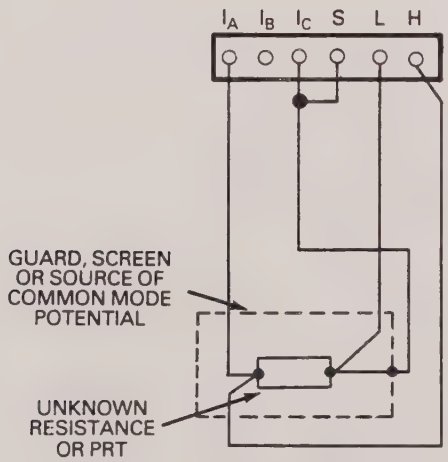
2.5 4-TERMINAL RESISTANCE AND PRT MEASUREMENT

2.5.1 Theoretical Diagram

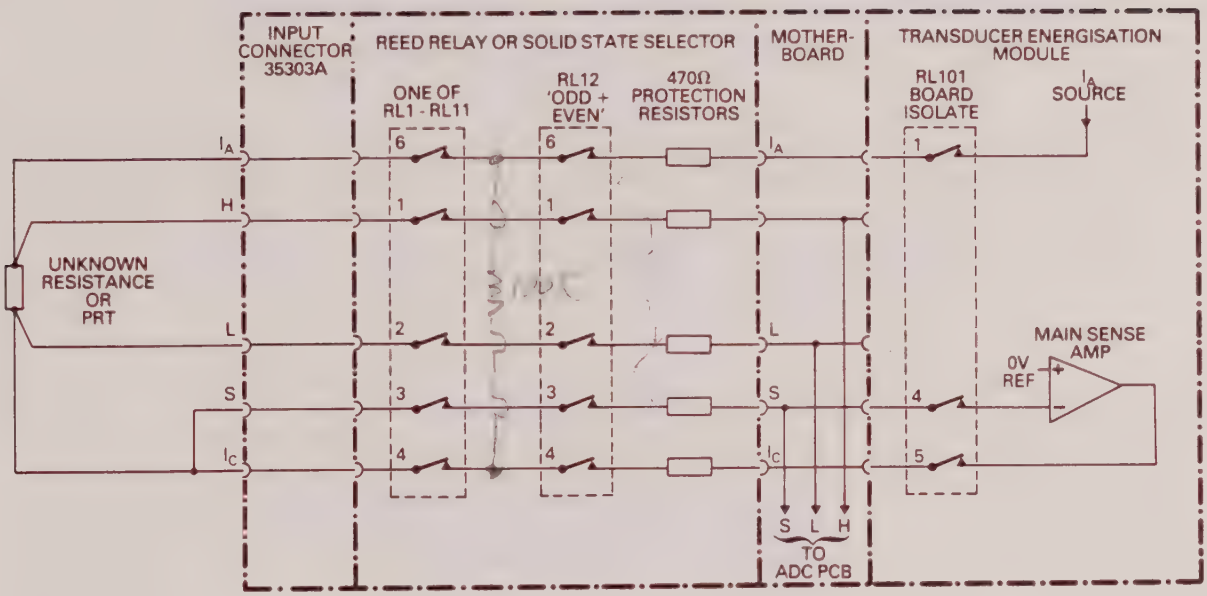


The energising current is generated and switched automatically, no additional linking or connections are required.

2.5.2 Practical Connection Diagram

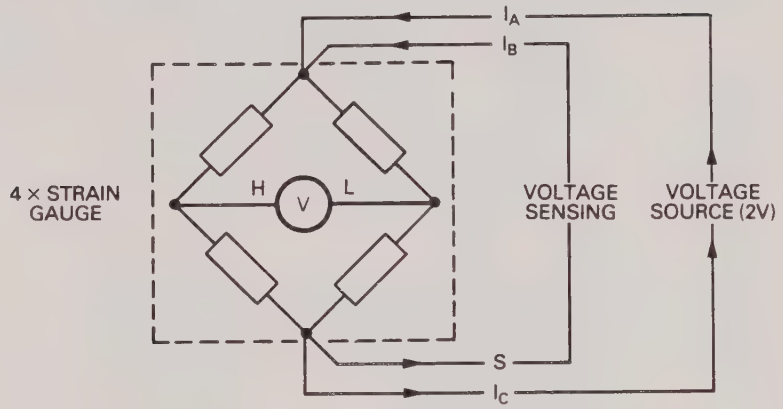


2.5.3 Internal Signal Organisation



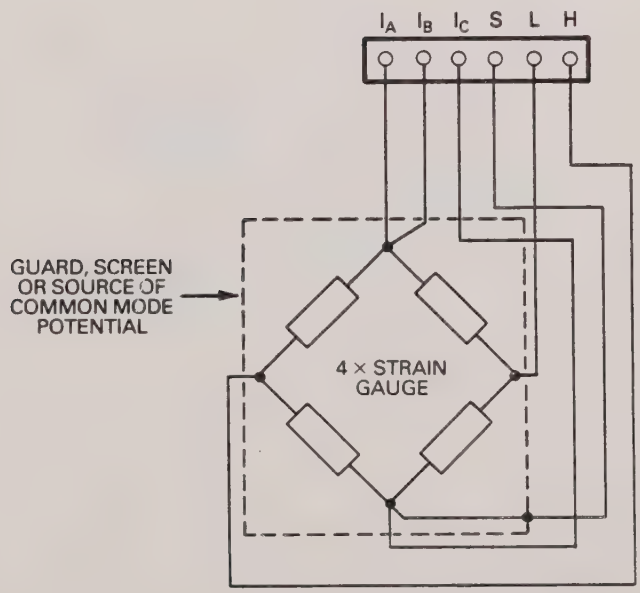
2.6 FULL BRIDGE STRAIN GAUGE MEASUREMENT

2.6.1 Theoretical Diagram

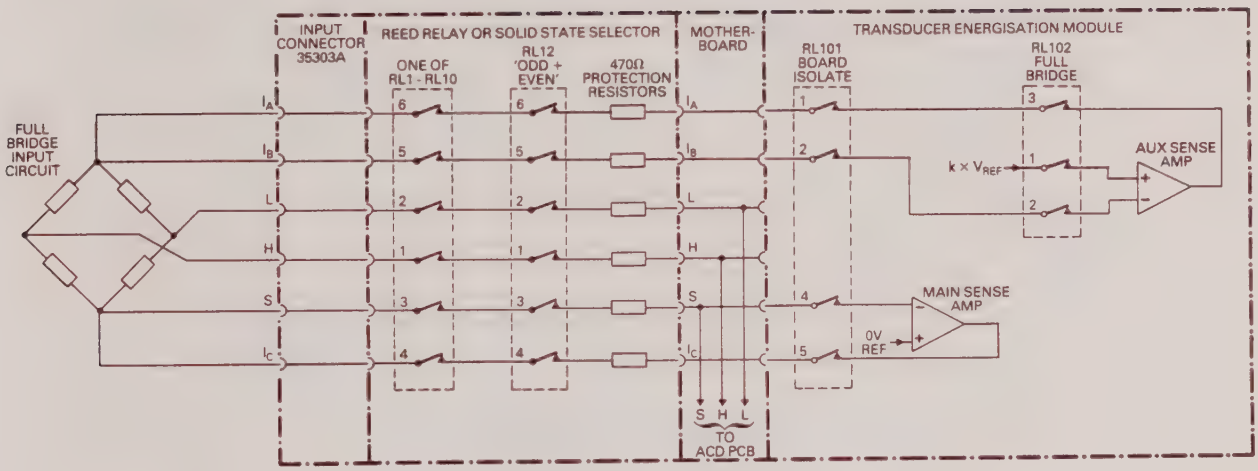


The energising voltage is generated and switched automatically, no additional linking or connections are required. Although the terminals used are the same as for current energisation, this configuration uses voltage energisation (2V).

2.6.2 Practical Connection Diagram



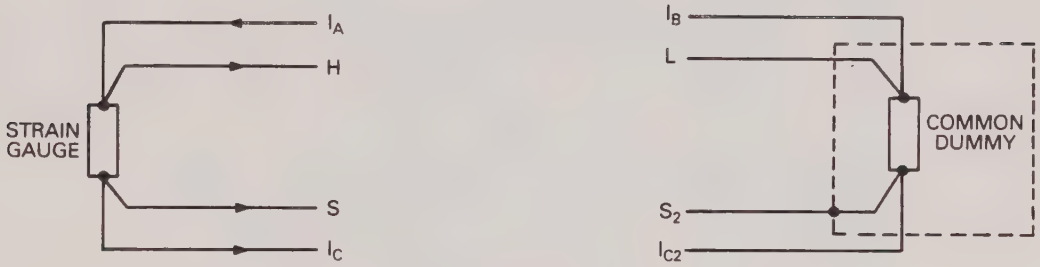
2.6.3 Internal Signal Organisation



2.7 ¼ BRIDGE STRAIN GAUGE MEASUREMENT

2.7.1 Theoretical Diagram

STRAIN GAUGE *DUMMY*

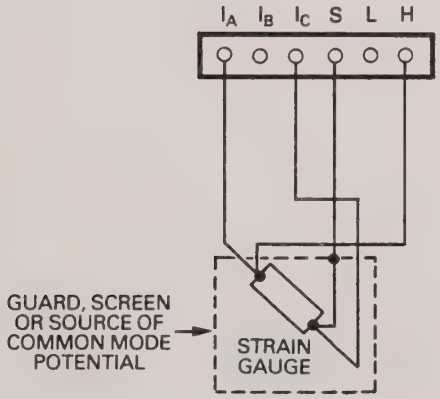


The energising currents are generated and switched automatically, no additional linking or connections are required.

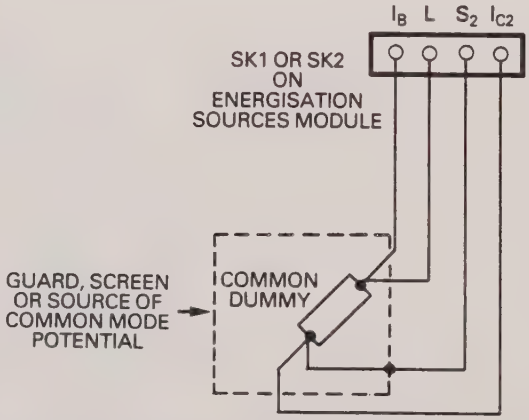
2.7.2 Practical Connection Diagram

STRAIN GAUGE *DUMMY*

NB. See procedure on page 4.30

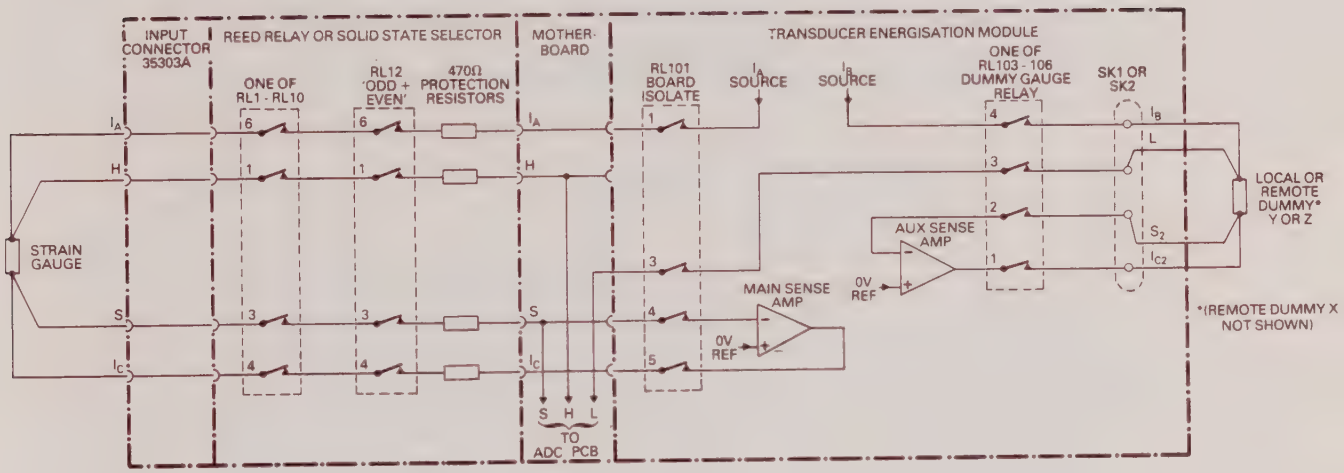


¼ bridge strain gauge configurations necessitate the use of a (shared) dummy gauge which is switched automatically by the logger to the strain gauge channel being measured. See below for further details and fitting instructions.



The diagram above shows the necessary connections when fitting a dummy provided by the user. A built-in 120Ω remote dummy is fitted (as standard) to the Energisation Sources Module; if this dummy is used no additional connections are necessary.

2.7.3 Internal Signal Organisation for ¼ bridge strain gauge with local or remote dummy



Fitting dummy gauges for ¼ bridge strain measurement

The dummy gauges necessary for ¼ bridge strain measurement are contained on the Transducer Energisation Module. One 120Ω remote dummy (Rem 120) is fitted as standard. There is provision for the user to fit 3 additional local or remote dummies, as shown in Fig 4.8:

User X	remote dummy only
User Y	local or remote
User Z	local or remote

'Remote' refers to a dummy which is remote from the gauge, such as one physically located on the Module. 'Local' refers to a dummy which is local to the gauge, i.e. located in close proximity to the gauge, and the connecting leads taken back to terminals on the Module. Local dummies can be used to provide compensation for drift of gauge resistance due to temperature.

Any one dummy can be used with any number of (¼ bridge) gauges, providing that the dummy resistance and the nominal gauge resistance are similar, and that the gauges all require a local or a remote dummy. Any dummy is automatically switched to any selected gauge by means of the internal switches.

The user may select the dummy required for each ¼ bridge strain channel using 'Channel Definition, Select Dummy: Rem 120, User X, User Y or User Z'.

To fit a dummy gauge to the Transducer Energisation Module carry out the following procedure:

1. Switch off logger.

2. Remove the Transducer Energisation Module from slot 14 in the mainframe, by turning the two ¼ turn locking bolts anti-clockwise and withdrawing the Module.

3. See Fig 4.8 for location of terminals to which dummies X, Y and Z should be connected.

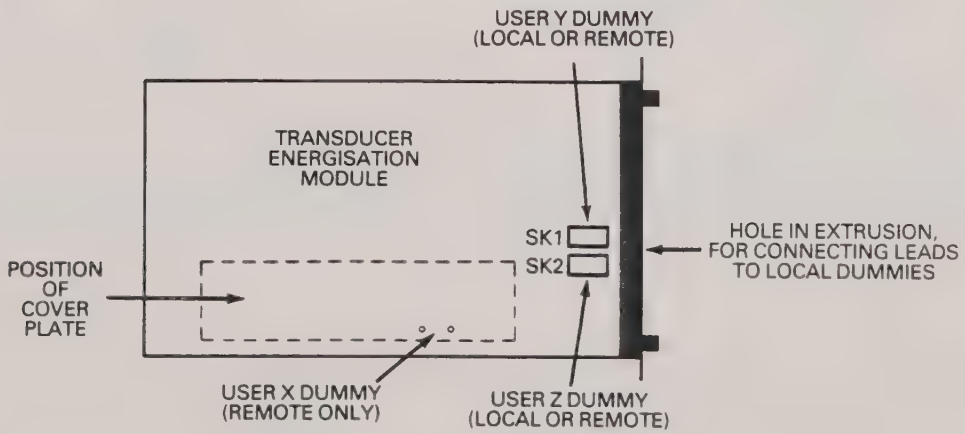
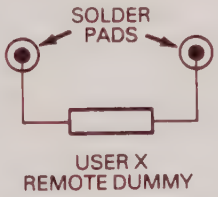


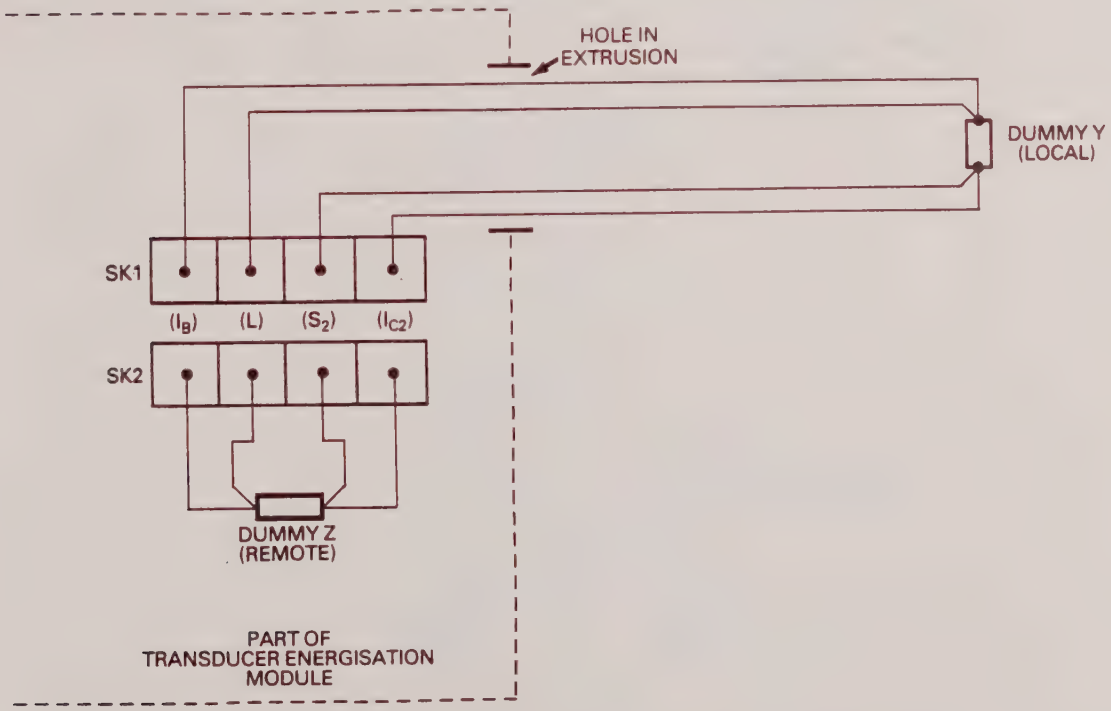
Fig 4.8 *Location of terminals for User X, Y and Z dummies*

4. To fit User X dummy, remove the cover plate on Module and solder the dummy between the two solder pads indicated in Fig 4.8. Replace cover plate.



The dummy resistance should be similar to the nominal resistance of the associated gauge(s).

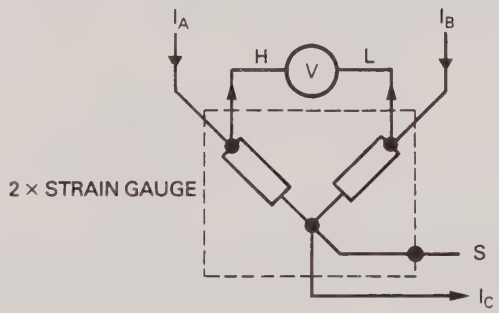
5. To fit User Y or User Z dummies, connect to SK1 or SK2 as shown below. Note that either or both of these dummies can be used as local or remote; in the diagram below dummy Y is shown as local and dummy Z is shown as remote. For local dummies, pass the leads connecting the dummy to SK1 or SK2 through the hole provided in extrusion; locate the local dummy close to the associated gauge(s).



6. Replace Transducer Energisation Module in slot 14 and lock Module using the two locking bolts.

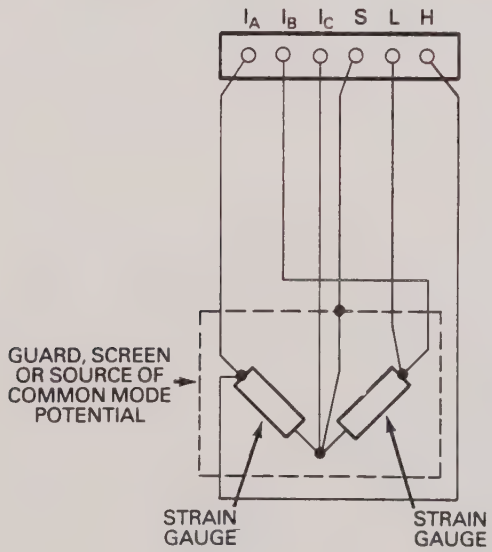
2.8 ½ BRIDGE STRAIN GAUGE MEASUREMENT

2.8.1 Theoretical Diagram

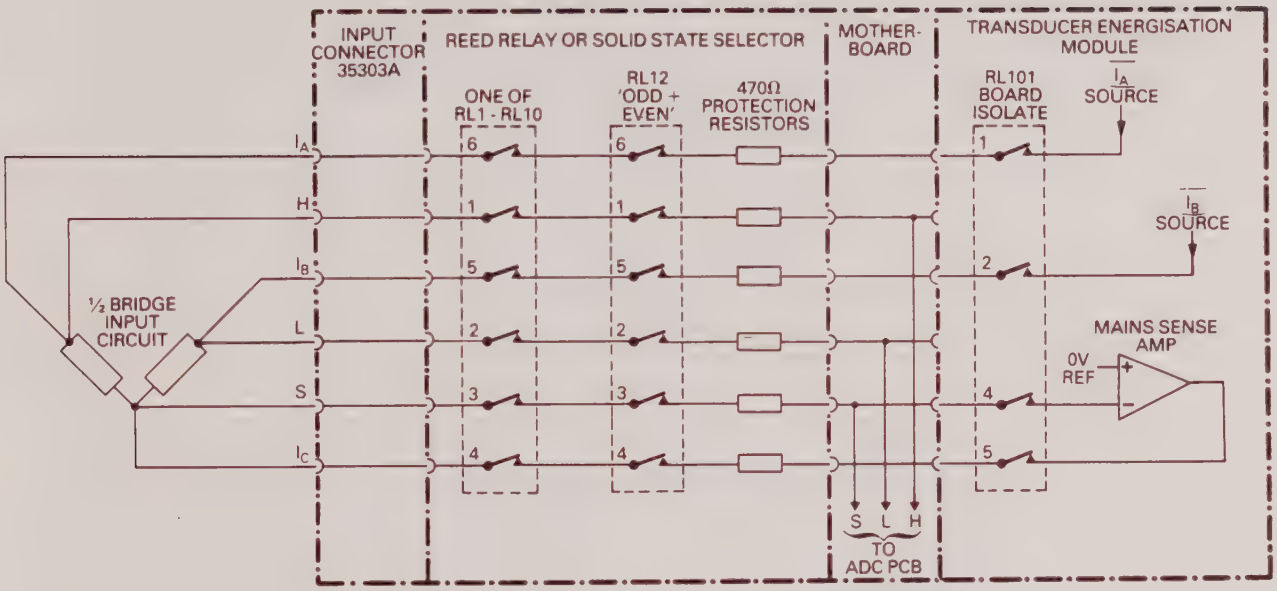


The energising currents are generated and switched automatically, no additional linking or connections are required.

2.8.2 Practical Connection Diagram

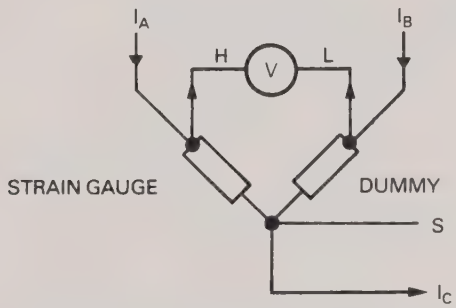


2.8.3 Internal Signal Organisation



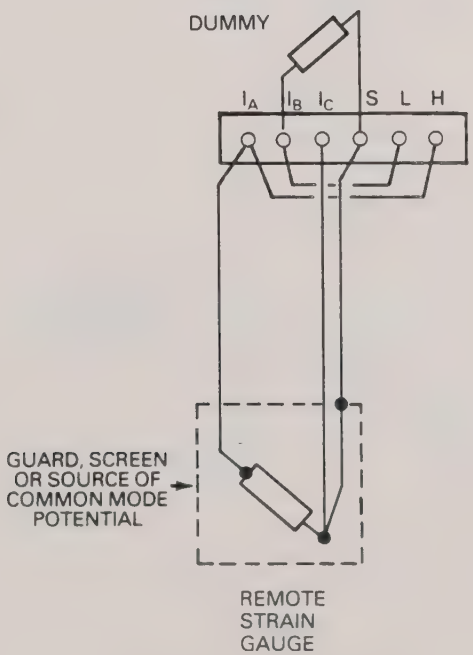
2.9 3-WIRE STRAIN GAUGE MEASUREMENT

2.9.1 Theoretical Diagram

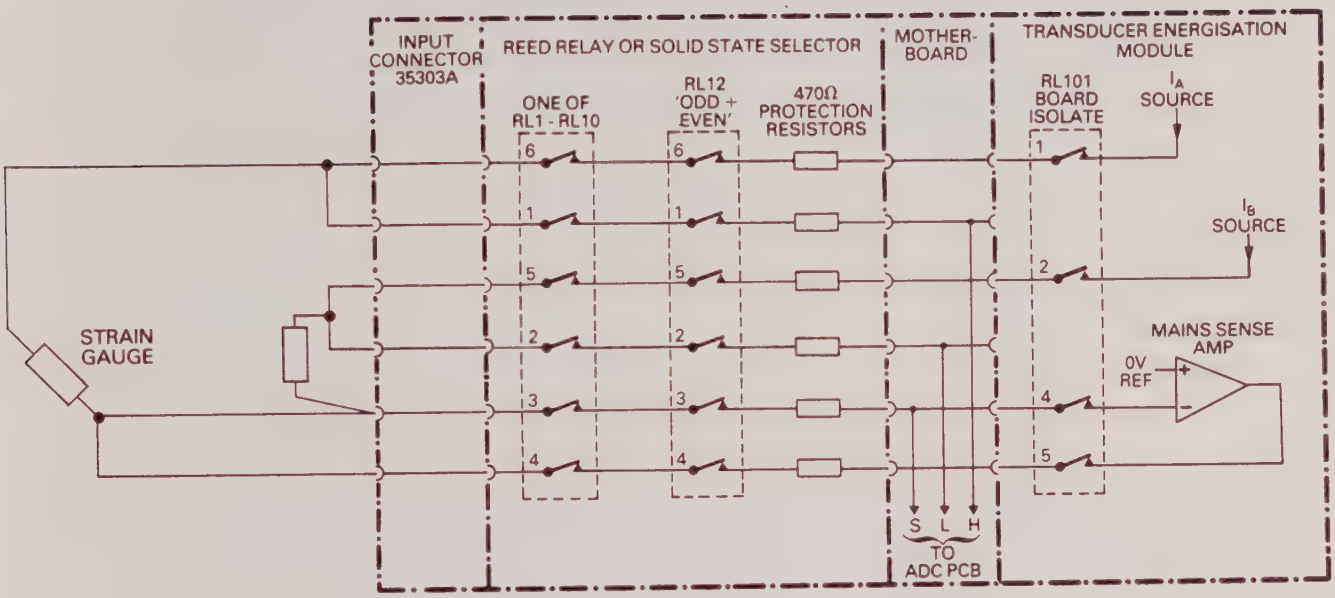


The energising currents are generated and switched automatically, no additional linking or connections are required.

2.9.2 Practical Connection Diagram



2.9.3 Internal Signal Organisation



Using the '3-wire' connection, the benefits of dual current energisation are retained, and the wiring is simpler than that needed for full bridges. The electrical arrangement is similar to that for '½ bridge' except that the dummy is connected directly across the terminals on the Input Connector.

Connection and Setting-up Procedure

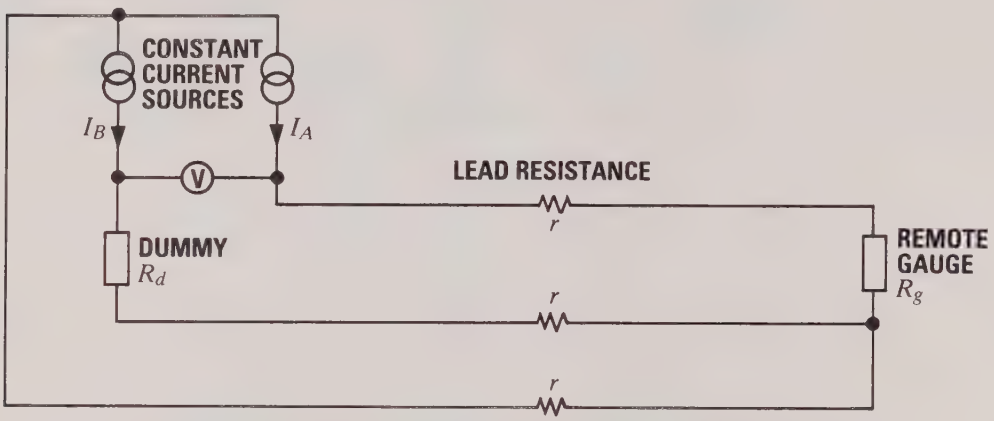
1. For each channel connect a dummy resistor and wiring as shown above. The dummy should be a value equal to the nominal gauge resistance. A miniature type (0.1W) wirewound may be used and will provide the best stability; but for less precise work metal film may be suitable.

2. For Channel Definition, select ½-bridge.

3. Refer to 'Temperature Compensation' and 'Gauge Sensitivity' on pages 4.36 and 4.37 before using the gauge(s).

Temperature Compensation

The diagram below shows the lead resistances in the three wires to the gauge. The lower one is in the return path for the constant currents and has no effect. The other two are in opposite arms so that changes in lead resistance due to temperature will balance out – provided the leads are of the same size and approximately equal in length.



Compensation is of importance when long leads are used because of the large temperature coefficient of copper. Expressed in terms appropriate to strain gauges, it is roughly 4000$\mu\Omega$ per ohm per °C. In other words, for a lead of 1Ω resistance the change due to temperature will be 4000$\mu\Omega$/°C. The change for 1μ strain in a 120Ω gauge is only 240$\mu\Omega$ (gauge factor = 2).

Whether the lead resistance is of significance obviously depends on the installation. As a guide, 100 feet (30m) of 7/.0076 (7 × 0.2mm) copper wire measures 2.5Ω. So a change of 10°C would cause a lead change of 100mΩ.

Gauge Sensitivity

Because lead resistance is in series with the gauge, resistance change due to strain (ΔR) now occurs in a total resistance of gauge-plus-lead ($R + r$). So the sensitivity of the gauge, represented by the gauge factor (G), is thereby reduced.

The usual gauge equation is:

$$\frac{\Delta R}{R} = G \times \text{strain}$$

With lead resistance this becomes:

$$\frac{\Delta R}{R + r} = \alpha G \times \text{strain}$$

where α is a factor less than unity.

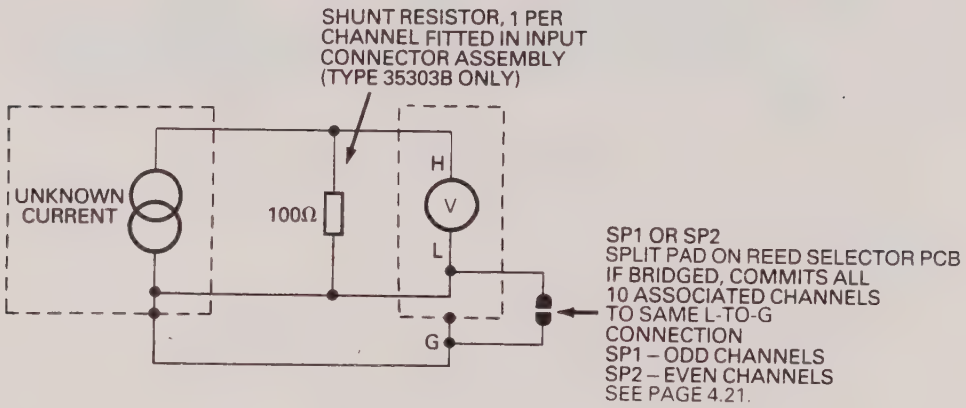
For the lead lengths discussed above the lead resistance is about 2% of the gauge resistance. So if G is used as quoted for the gauges the error in absolute strain would also be about 2%. But the value of G given by a supplier is unlikely to be better than 1%. So in most practical situations the use of a modified gauge factor αG will give only marginal improvement in absolute strain. Stability and repeatability will not be impaired.

The change of lead resistance with temperature will also affect sensitivity. Compared with the absolute lead resistance this is a second order term which can be ignored.

Most people who use strain gauges are less interested in absolute strain; a measurement of 5000μ strain to the nearest 100μ strain being more than adequate. The strain itself is often not the final parameter that is needed; stress, for instance, require modulus of elasticity to be used which may not be known precisely. However, if there are circumstances when an improved absolute accuracy is essential, calibration can be carried out to obtain αG. The calibration procedure is described in Solartron Publication Ref. No. 001/83 '3-Wire Strain Gauges With Orion.'

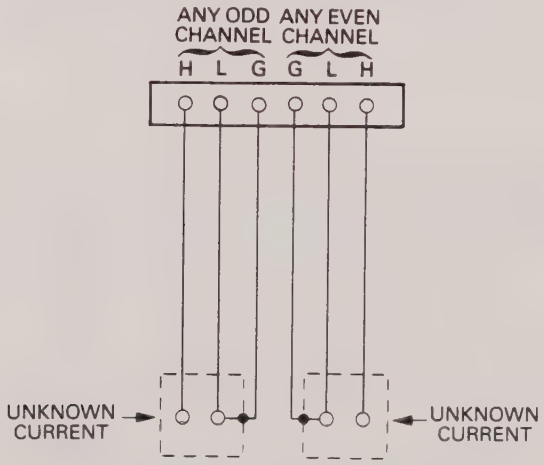
2.10 3-POLE CURRENT MEASUREMENT

2.10.1 Theoretical Diagram


SHUNT RESISTOR, 1 PER CHANNEL FITTED IN INPUT CONNECTOR ASSEMBLY (TYPE 35303B ONLY)
UNKNOWN CURRENT
100Ω
H
V
L
G
SP1 OR SP2
SPLIT PAD ON REED SELECTOR PCB
IF BRIDGED, COMMITS ALL 10 ASSOCIATED CHANNELS TO SAME L-TO-G CONNECTION
SP1 – ODD CHANNELS
SP2 – EVEN CHANNELS
SEE PAGE 4.21.

2.10.2 Practical Connection Diagram

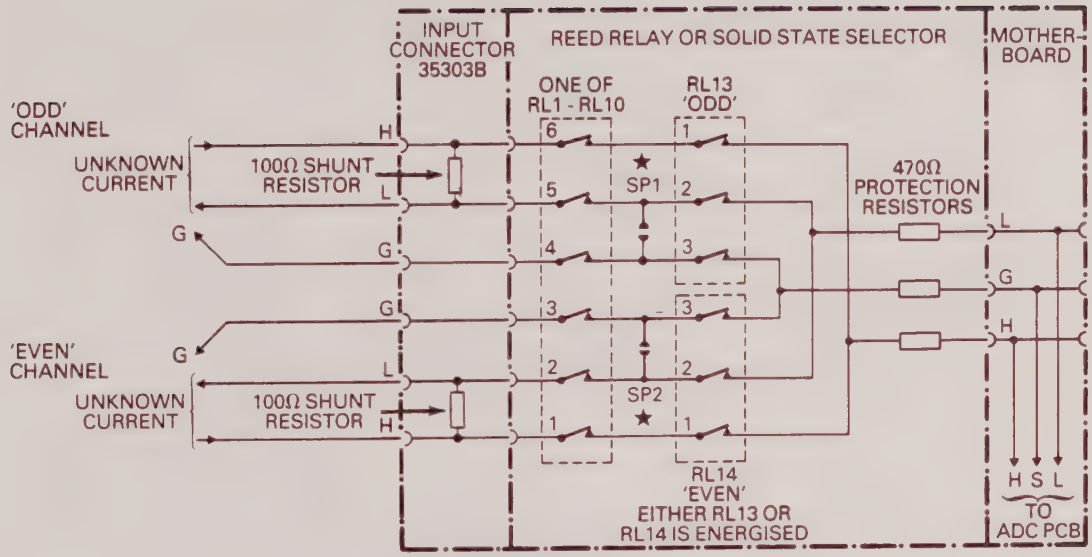
Input Connector 35303B has internal shunt resistors fitted for current measurement; a 100Ω resistor is fitted across each channel (between H and L).



CONNECT G TO GUARD, SCREEN OR SOURCE OF COMMON MODE POTENTIAL, OR CONNECT G TO L, AT POINT OF MEASUREMENT OR USING SPLIT PAD IN LOGGER.

Current measurements can be made using Input Connector 35303A. When using a 35303A fit a shunt resistor (e.g. 100Ω) between H and L terminals on each current channel, select 'voltage' measurement for each current channel and then use CONVERSION to convert the measured voltage into current.

2.10.3 Internal Signal Organisation



★ NOT ON SOLID STATE SELECTOR

Chapter 5
Using the Event-Status Input Module (35302A)

Contents

		Page No.
1	General Information	5.2
1.1	Introduction	5.2
1.2	Status Inputs	5.3
1.3	Event Inputs	5.4
1.4	Counting and Totalising Events	5.5
1.5	Setting up Alarms for Status and Event Inputs	5.5
1.6	Channel Numbering and Pin Assignments for Inputs	5.7
1.7	Input Specifications	5.10
1.8	Input Energisation	5.11
2	Input Connections	5.12
2.1	General Procedure	5.12
2.2	Fitting additional capacitors for greater contact-bounce filtering	5.17

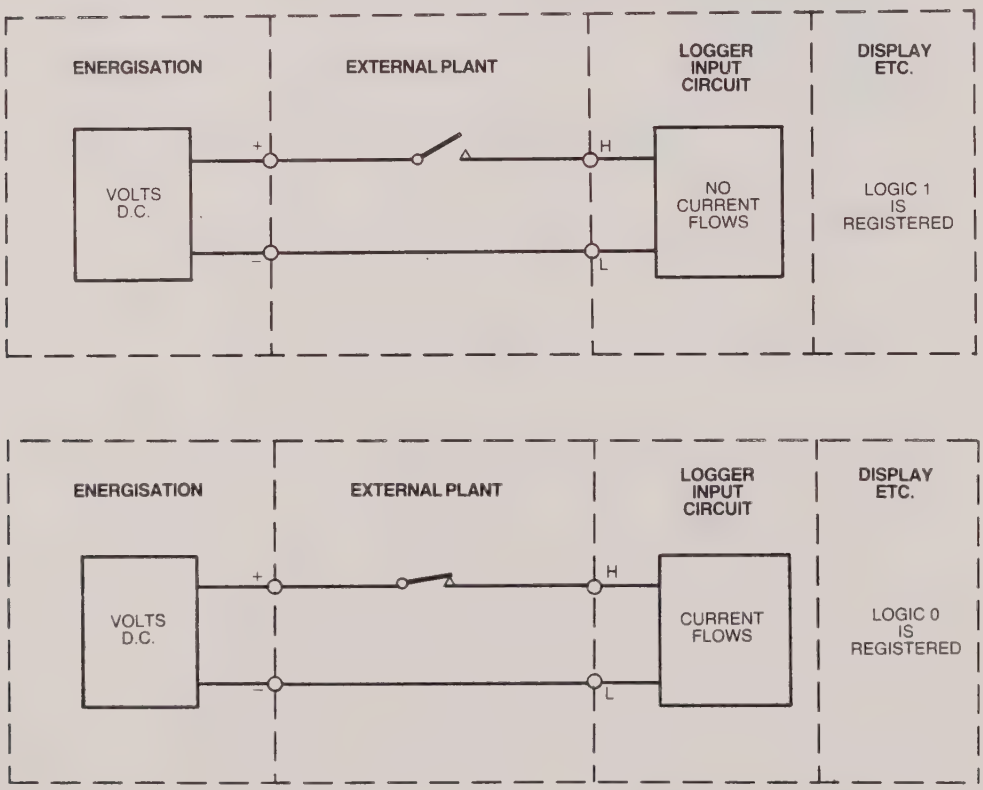
1 GENERAL INFORMATION

1.1 INTRODUCTION

The Status-Event Input Module 35302A enables the logger to respond to, and record, digital signals. A digital signal can be in only either of two "states", referred to here as Logic 0 and Logic 1. A switch is a good example of a "digital" device, as its "status" can only be either open or closed.

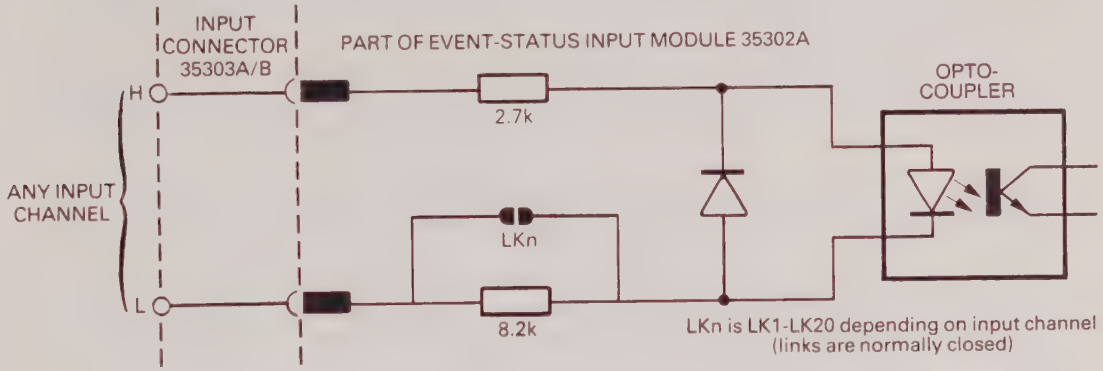
In the convention adopted for Orion, Logic 0 represents the "presence", and Logic 1 the "absence", of a current or voltage at the input. The actual voltage levels which the logger recognises as a 1 or a 0 are defined in the Specification.

Input Circuit Each module will accept 20 digital inputs. Each input can be visualised in the simplified form shown in the Input Circuit Schematics below, with the energisation voltage derived from either the external plant, or from the logger, as described later.



Input Circuit Schematics

Each input consists of an independent pair of terminals which are opto-isolated and fully floating, as shown below:



Input Circuit for each channel

Each of the 20 inputs may be used as either a Status or an Event Channel. These two distinct types are described in the following Sections.

1.2 STATUS INPUTS

Status Channels record the **state** of the inputs, i.e. whether they are at Logic 0 or Logic 1. The inputs are scanned at the times and intervals defined in a Scan Task. In scanning the status inputs, the logger observes the input voltage level, e.g. 0V or 5V. The scanning speed is approximately 500 channels/s. Status inputs can be used as individual channels, or they can be grouped.

The logic state (1 or 0) detected for each separate status input channel will be dependent on whether the input circuit is conducting or not. If conducting, a logic 0 state is detected; if not conducting a logic 1 state is detected.

The alarm state of any individual channel can be defined as 1 or 0. Alternatively, a number of consecutive channels (maximum of 8) can be grouped, and a group state can be defined as an alarm. A typical group state would be 1101 for a 4-channel group. (Further details on alarms are given in section 1.5, page 5.5).

BCD channels Status inputs can be used for reading bcd (binary coded decimal) decades in 8-4-2-1 form. Normal bcd or inverted bcd can be input; this is selected as part of the Channel Definition, see Chapter 9. For example, the number 9, (i.e. 8 + 1), can be represented in bcd form as either:

	8	4	2	1
Normal bcd; the required digits are represented by Logic 0, or:	0	1	1	0
Inverted bcd; the required digits are represented by Logic 1	1	0	0	1

Each bcd decade uses 4 consecutive input channels, the lsb (least significant bit) is the lowest Channel number, the msb (most significant bit) is the highest Channel number.

e.g. If channels 41 to 44 are used, the inputs are connected thus:

Channel numbers	44	43	42	41
bcd significance	8	4	2	1

Successive decades may be connected in a similar fashion:

Channel Numbers	52	51	50	49	48	47	46	45	44	43	42	41
bcd significance	800	400	200	100	80	40	20	10	8	4	2	1
	hundreds decade				tens decade				units decade			

A maximum of five decades may be connected using one Event-Status Input Module, or up to six decades may be connected if they are distributed across two Modules.

Note that when a group of 4 channels is used as a bcd decade (e.g. 41 to 44) the whole group is referred to by the lowest channel number in the group (e.g. 41). Similarly, if channels 41-60 are used for five bcd decades, only channel 41 is declared in the channel definition, the logger will automatically assign the rest of the channels in sequence.

For each bcd decade, the logic levels for the 4 separate input channels are detected. These are converted by the logger into a decimal number, as shown below in **inverted** bcd.

Typical channel numbers	52	51	50	49	48	47	46	45	44	43	42	41
Typical measured logic levels	1	0	0	1	0	1	1	0	0	1	0	1
Decimal values	800	0	0	100	0	40	20	0	0	4	0	1
	= 900				= 60				= 5			

∴ decimal equivalent of 100101100101 bcd is 965.

Alarm values for bcd channels are defined using decimal values. The alarm conditions which can be applied to bcd channels are the same as those used for the analogue channels.

1.3 EVENT INPUTS

Event Channels record **transitions** from one logic state to the other, i.e. from Logic 1 to Logic 0, or from Logic 0 to Logic 1, or both, as requested by the user. Logic 0 and Logic 1 are defined in the Introduction to this Chapter. When selecting the Event menu, the Display refers to the transitions as:

"lo-to-hi" which represents Logic 0 to Logic 1
"hi-to-lo" which represents Logic 1 to Logic 0
"any" which represents either of the above

If "any" is selected, then either a lo-to-hi or a hi-to-lo transition will be regarded as an Event. This could be used, for instance, to detect both the opening and closing of a switch, i.e. when the switch opens an Event occurs and when the switch closes an Event occurs.

The inputs are monitored continuously, whilst the Task containing them is active, and an Event is detected as soon as it occurs. All events are regarded as alarms. Any event can be used as the triggering source for another task which could then, for example, measure other input channels, or alternatively, control the digital outputs (on the Digital Input/Output Module, 35302B).

A single channel can detect events up to a maximum rate of 400 transitions/second. If more than one channel is being used to observe events, ORION can handle all the event inputs up to a total (the aggregate from all channels) of 400 transitions/second.

Note that any event input *pulse* actually consists of two transitions:

lo to hi and hi to lo, or
hi to lo and lo to hi.

Both transitions in either case contribute to the total detected by the processor, even though only the lo to hi, or the hi to lo, transition is selected for processing/alarm action.

The time at which an event occurs is measured to an accuracy of 2ms. (The minimum pulse length for an event input is 2ms, i.e. it must remain in the hi, or lo, state for not less than 2ms to be detected. (The total event processing time is 2ms).

1.4 COUNTING & TOTALISING EVENTS

In addition to simply responding to individual signal transitions, the transitions can also be counted. Count totals are measured and output only when that channel is scanned; channels used to count events must therefore be incorporated in a scanning-type task (like analogue channels).

ORION can be set up to give two types of event counting:

Totalising: the cumulative total of transitions since the start of a task is produced; i.e. the total from each new scan is added to the previous total. The maximum count is 65535.

Counting: the total count of transitions since the last reading is produced; i.e. the counter is reset to zero after each scan.

Totals from either type of event counting can have conversion equations applied to give readings in practical units. For example, the total transitions counted over a specified period (as determined by the task scanning rate) could be a count of items observed on a moving conveyor belt. The number of items observed over a 10 second period could be multiplied by 360 to give a rate of 'items per hour'.

1.5 SETTING UP ALARMS FOR STATUS & EVENT INPUTS

STATUS INPUTS

(a) Single Channels Consider the simple case of one status input channel with a switch connected as shown below.

SWITCH POSITION	MEASURED LOGIC STATE
open	1
closed	0

If the switch is open, no current flows in the input circuit; when the channel is scanned, a logic 1 state is detected. If the switch is closed, current flows in the input circuit; when the channel is scanned, a logic 0 state is detected. This establishes the input conditions for a measured result of logic 1 or 0.

To set up an alarm condition for this channel the user must select Match or Mismatch, and a logic state.

Alarm on Match: Select 1 or 0, this represents an alarm level of logic 1 or 0 respectively. When the detected logic level is the same as the level selected (i.e. the levels 'Match') then an alarm will occur. No alarm occurs if the levels are different.

Alarm on Mismatch: Select 1 or 0, this represents an alarm level of logic 1 or 0 respectively. When the detected logic level is different from the level selected (i.e. the levels 'Mismatch') then an alarm will occur. No alarm occurs if the levels are the same.

The Match/Mismatch level is known as a 'pattern', the reason for this will become evident after the reader has considered the alarm condition for grouped status channels, see below.

(b) Grouped channels As explained in a previous section, status inputs may be used individually or as a group of up to 8 channels. For simplicity, it is normal for any group to consist of consecutive channels, e.g. channels 21 to 28 would be used as an 8 channel group. The whole group is referred to by the lowest channel number only, e.g. 21.

When a group of channels is scanned, the output from each channel is detected and the result for the whole group is output as a pattern of logic 1's and 0's. For example, the 8 channel group referred to as channel 21 might produce the following result:

Ch 21 00110010

This corresponds with the separate channels in the group as follows:

Channel Number	28	27	26	25	24	23	22	21
Logic Level Detected	0	0	1	1	0	0	1	0

Note that the lowest numbered channel is always presented on the right, the highest on the left. The significance of all logic 1's or 0's detected is identical to that described under 'Single Channels' above.

To set up an alarm condition for grouped channels the user must select pattern Match or Mismatch, and a pattern of logic 1's and 0's (or ⊡ for 'dont care' channels, see below).

Alarm on Match: Select a logic 1 or 0 for each channel in the group, e.g. for an 8 channel group set up a total of 8 logic 1's or 0's; this could typically be: '00010001'. This represents an alarm pattern. When the measured pattern is the same as the alarm pattern selected (i.e. all 'bits' in the pattern 'Match') then an alarm will occur. No alarm occurs if the patterns are different.

e.g. 1	Measured pattern	01010000	
	Alarm pattern selected	00010001	= No alarm
e.g. 2	Measured pattern	00010001	
	Alarm pattern selected	00010001	= ALARM condition

Alarm on Mismatch: Select a logic 1 or 0 for each channel in the group, e.g. for an 8 channel group set up a total of 8 logic 1's and 0's, this could typically be '11100001'. This represents an alarm pattern. When the measured pattern is different from the alarm pattern selected (i.e. the patterns 'Mismatch') then an alarm will occur. No alarm occurs if the patterns are identical.

e.g. 1	Measured pattern	01010000	
	Alarm pattern selected	01010000	= No alarm
e.g. 2	Measured pattern	01000000	
	Alarm pattern selected	01010000	= ALARM condition

Note that an alarm pattern, for Match or Mismatch, is entered from the keypad as follows:

the bit corresponding with the highest numbered channel in the group is entered first,
the bit corresponding with the lowest numbered channel in the group is entered last.

Thus if alarm pattern 00001111 is entered for channels 21 to 28 the bits correspond with the channels as follows.

Selected pattern	0	0	0	0	1	1	1	1
Corresponding channel numbers	28	27	26	25	24	23	22	21

Dont-care This is a masking facility which allows the user to make any of the bits in an alarm pattern into a 'dont-care' state. When any bit is defined as 'dont-care' the detected level (Logic 1 or 0) of the corresponding channel will have no effect on the alarm state, but the level detected will be output at the output device in the normal way. To set up a dont-care state, simply enter [*] instead of a 1 or 0. For example, instead of the alarm pattern shown above (00001111) for channels 21 to 28, the user could enter 0000[*]111. This would mean that whatever state (Logic 1 or 0) was detected at channel 24, it would have no effect on the alarm condition. The detected state of all channels, including channel 24, would however be output to the output device in the normal way when the channels were scanned.

(c) bcd channels As the results from all bcd channels are converted to a decimal number, alarm conditions for bcd channels are the same as for analogue channels.

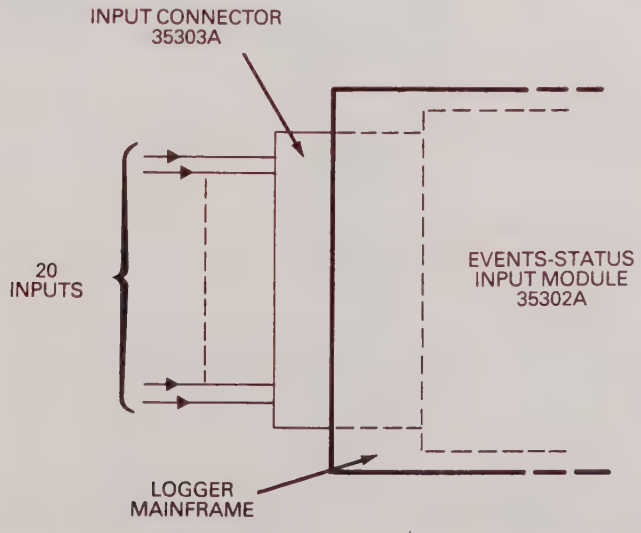
EVENT INPUTS

All events detected are regarded as alarms. An event is detected when a transition of logic state occurs on an event input channel. A valid event can be:

a transition from hi to lo only
a transition from lo to hi only
either transition

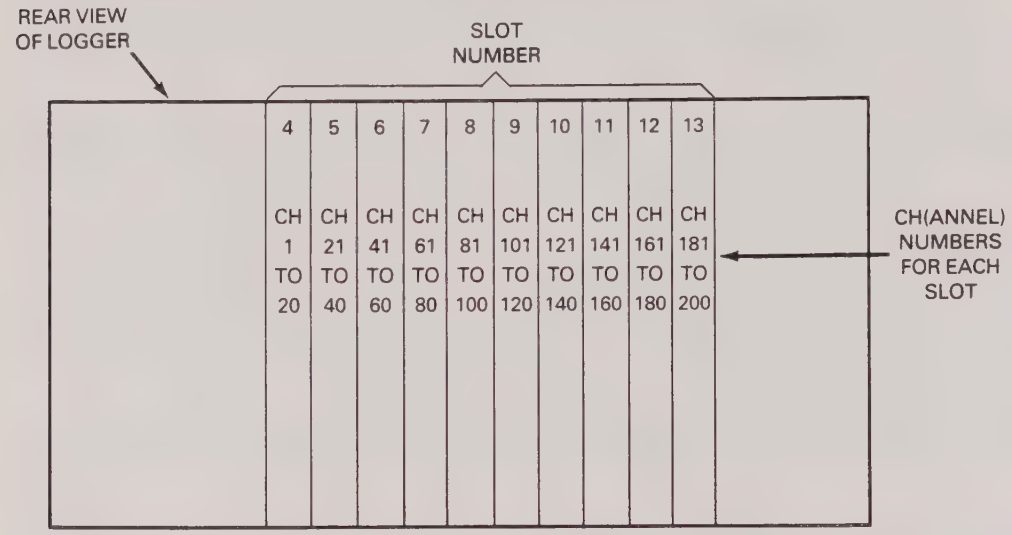
1.6 CHANNEL NUMBERING AND PIN ASSIGNMENTS FOR INPUTS

Inputs are connected to the logger using Input Connector 35303A which mates with the Event-Status Input Module in the logger mainframe.



One Event-Status Input Module can occupy any of the mainframe slots 4-13 inclusive. Channel numbering of the inputs is dependent on the location of the Module in the mainframe, see Fig 5.1.

The input connections are made to screw terminals on the input connector. A simplified diagram of the input connector with cover removed is shown in Fig 5.2, and the channel numbering and pin assignment for the terminals is shown in Fig 5.3.



An Input Connector and the associated Event-Status Input Module may be fitted in any of the slots 4-13 in the logger mainframe; channel numbering is, however, always dependent on position in the mainframe.

Fig 5.1 *Channel Numbering in logger mainframe.*

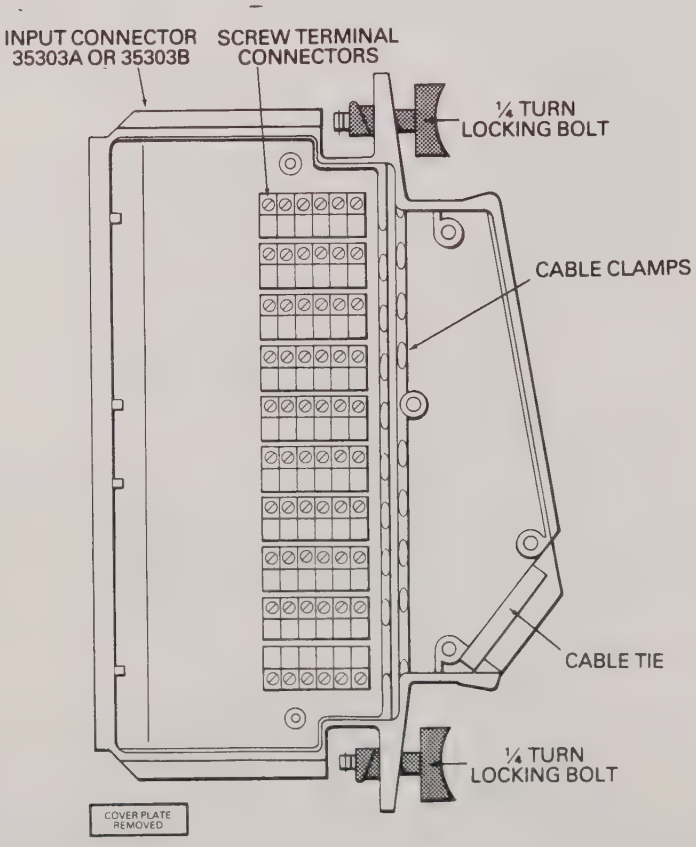
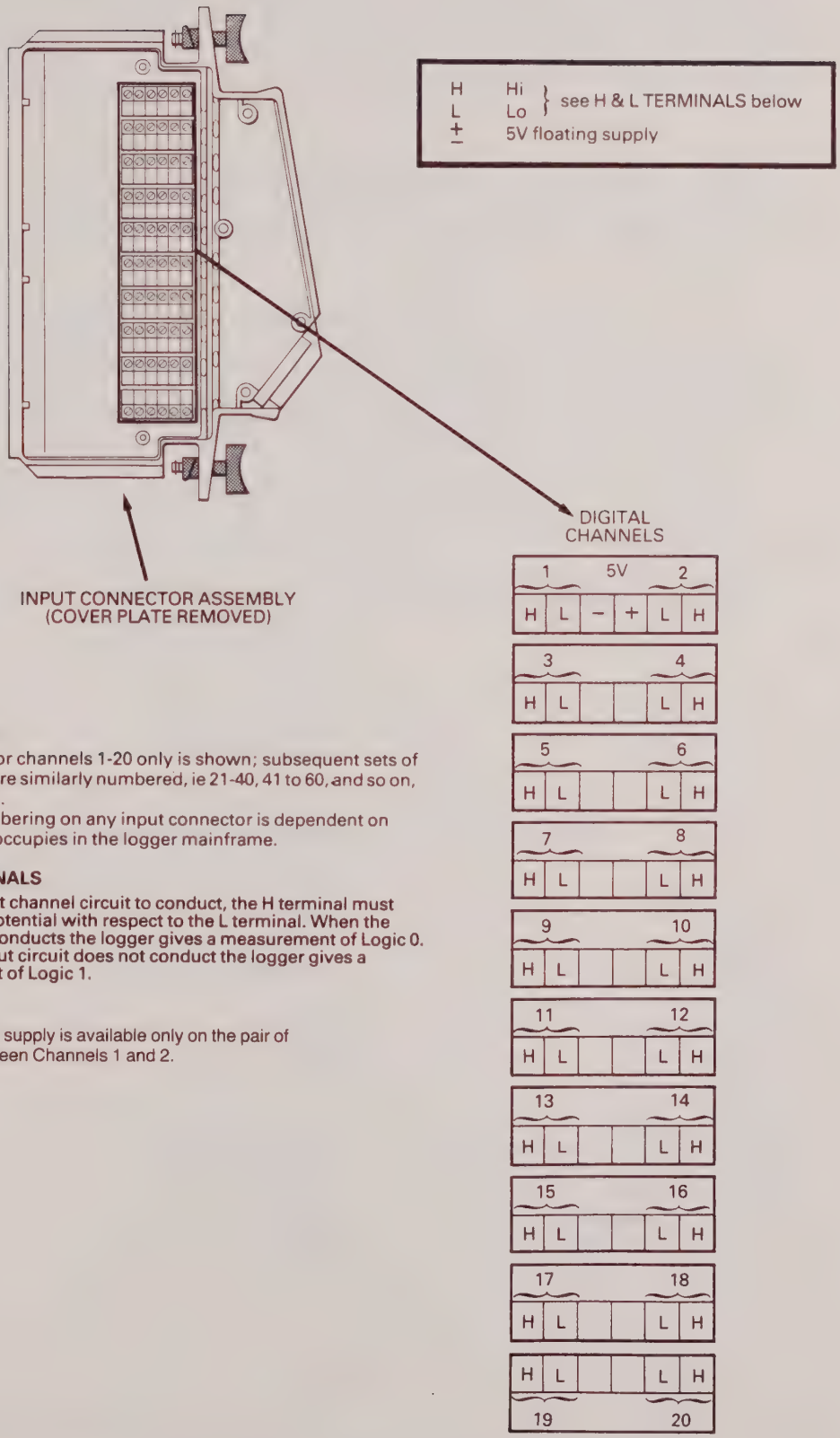


Fig 5.2 *Input Connector with cover removed.*



NUMBERING

Numbering for channels 1-20 only is shown; subsequent sets of 20 channels are similarly numbered, ie 21-40, 41 to 60, and so on, up to 181-200.

Channel numbering on any input connector is dependent on which slot it occupies in the logger mainframe.

H & L TERMINALS

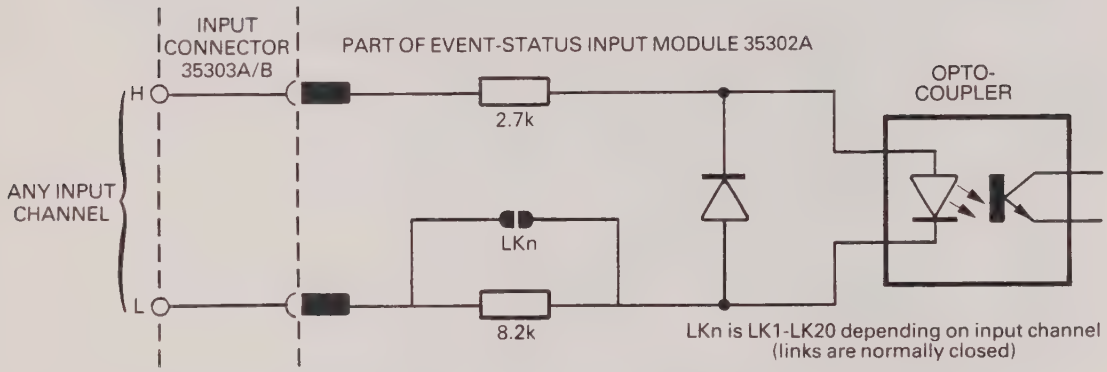
For each input channel circuit to conduct, the H terminal must have a +ve potential with respect to the L terminal. When the input circuit conducts the logger gives a measurement of Logic 0. When the input circuit does not conduct the logger gives a measurement of Logic 1.

Note:
The 5v floating supply is available only on the pair of terminals between Channels 1 and 2.

Fig 5.3 *Input Connector: Channel numbering and pin assignment.*

1.7 INPUT SPECIFICATION

Internal input circuit for each channel.



Number of input pairs:	20, opto isolated
Input voltage (H to L terminals):	
For logic 0 detected	+ 4V to + 12V, Lkn closed + 12V to + 80V, Lkn open
For logic 1 detected	< + 1V
Input current range:	0.7mA to 7mA
Maximum permitted common mode voltages:	> ± 200 with respect to earth > ± 200 between channels
Energisation source:	5V/40mA, ± 200V common mode
Event inputs	
Transitions recognised	Lo to Hi, Hi to Lo, or both
Event processing speed	400/s
Event totalising	65535 max.
Event timing	± 2ms
Status inputs	
Level recognised	Logic 1 or Logic 0
Channel grouping	single or grouped bits
Module interfacing	8-4-2-1 bcd
Reading rate, single bit	500 bits/s
as groups	1000 bits/s

1.8 INPUT ENERGISATION

Status/Event inputs must be at logic 1 or logic 0. Any one of the following devices can be connected to each input to give a Logic 0-to-Logic 1 signal: relay contacts, switch contacts or outputs from digital instrumentation. Energisation is required for each input regardless of whether it is defined as a status or event channel. Input energisation can be derived from any one of the following three sources:

1. The internal 5V supply (output at input connector terminals, see Fig 5.3). The internal supply provides sufficient current to power all 20 inputs.

2. One or more external power supply; see '1.7 Input Specification' for permitted input voltages etc.

3. The digital instrumentation or plant to which the inputs are connected.

All input channels energised by the same supply (internal or external) will have the same common mode voltage. Separate floating supplies must be used for any input (or group of inputs) which require a different common mode potential.

N.B. Channels 1 and 2 (only) must be at the same common mode potential as the internal power supply. To ensure this, if Channels 1 and 2 are energised from external plant one terminal of Channels 1 and 2 should be connected to one terminal of the internal supply.

Any mixture of the three types of energisation listed above can be used on the 20 input channels.

2 INPUT CONNECTIONS

2.1 GENERAL PROCEDURE

1. Remove the Input Connector from the logger mainframe by turning the two ¼-turn locking bolts anti-clockwise and withdrawing the assembly, see Fig 5.4.

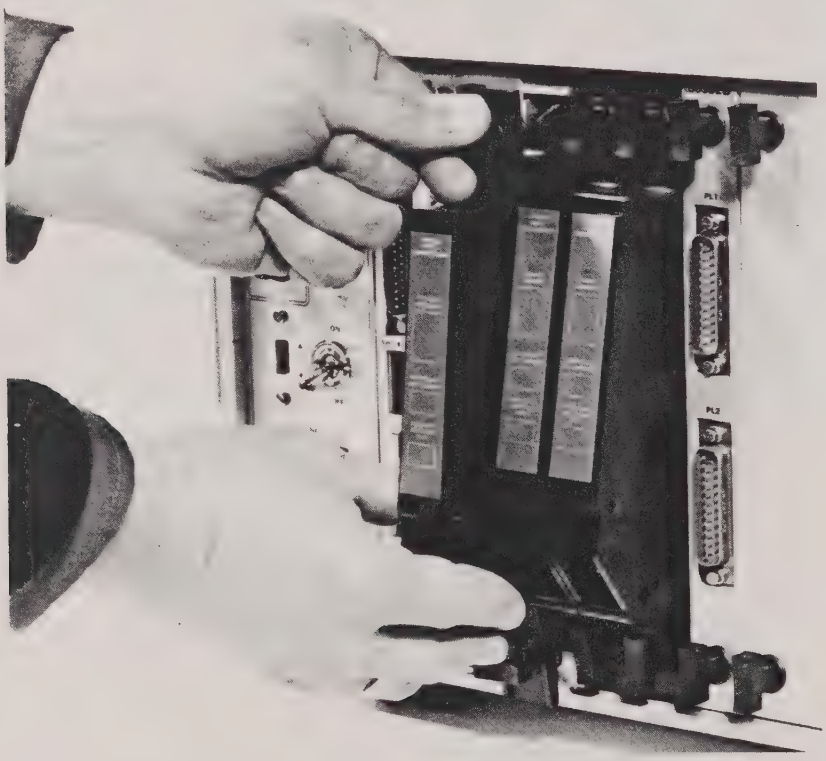


Fig 5.4 *Removing an Input Connector from the logger*

2. Remove the cover plate from the Input Connector to reveal the screw terminals.

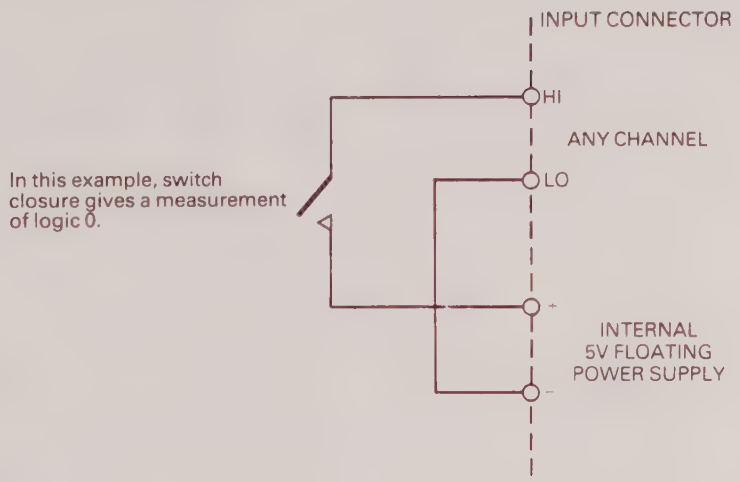
3. See Fig 5.1 and 5.3 to determine channel numbering and assignment of terminals.

4. Connect the digital inputs to the Input Connector as specified in instructions 4.1, 4.2 or 4.3 depending on the type of input and energisation, see list below.

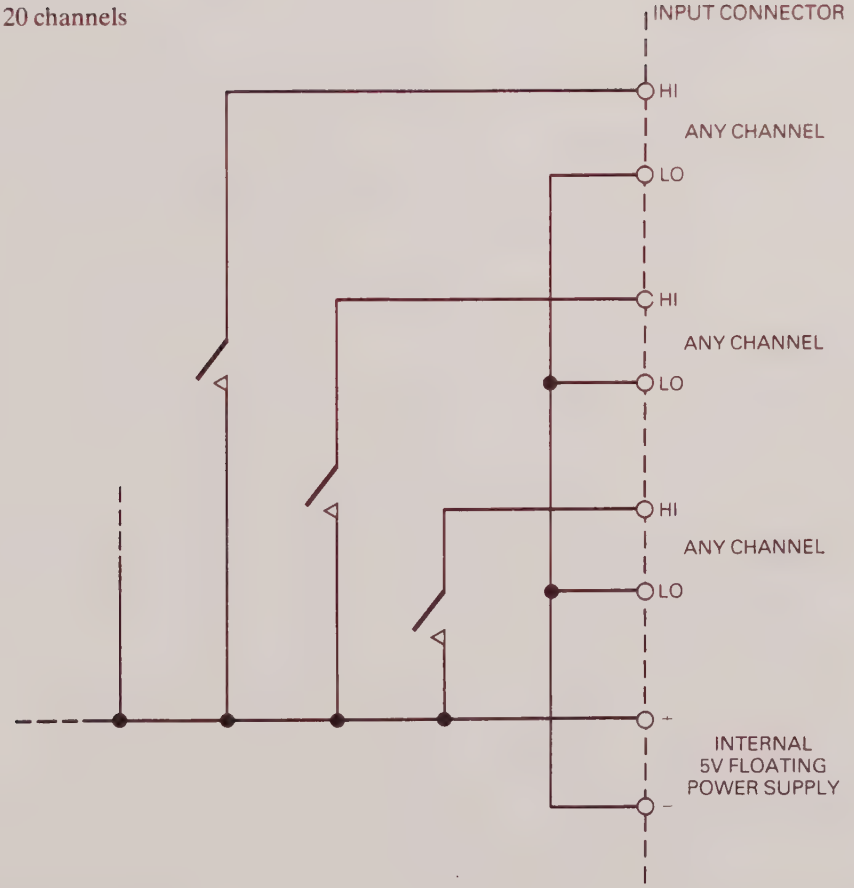
    4.1 Contact inputs using the internal floating supply.
    4.2 Contact inputs using an external supply.
    4.3 Outputs from digital instrumentation.

4.1 Contact inputs using the internal floating supply

(a) Single channel

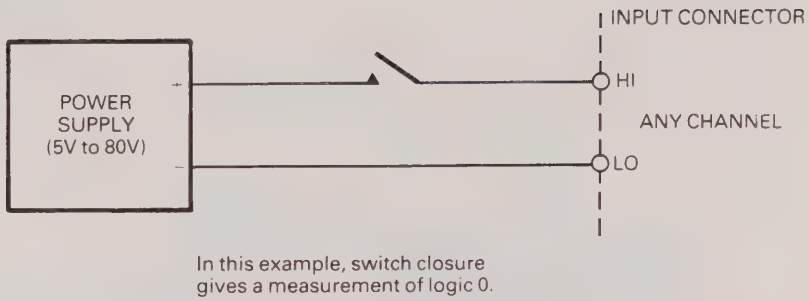


(b) Up to 20 channels



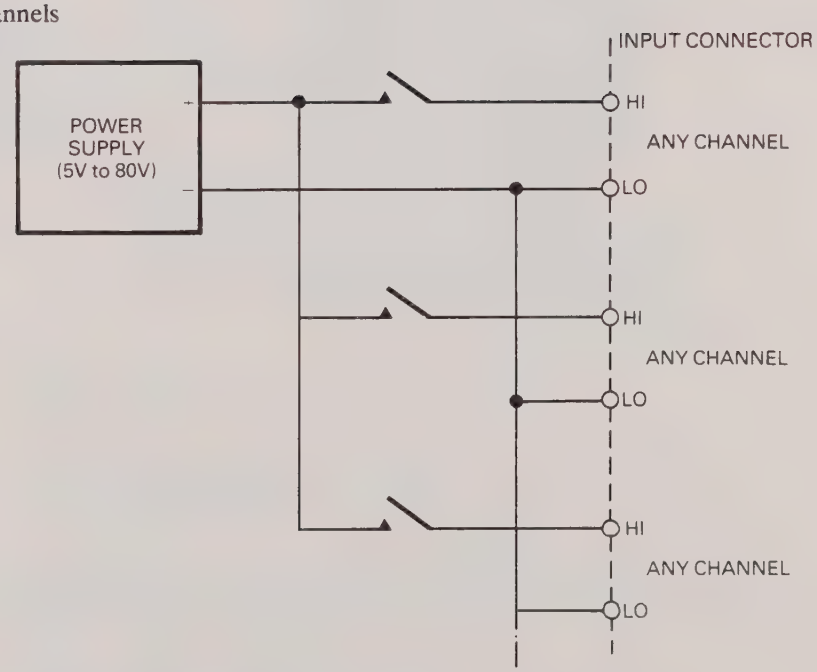
4.2 Contact inputs using an external supply

(a) Single channel



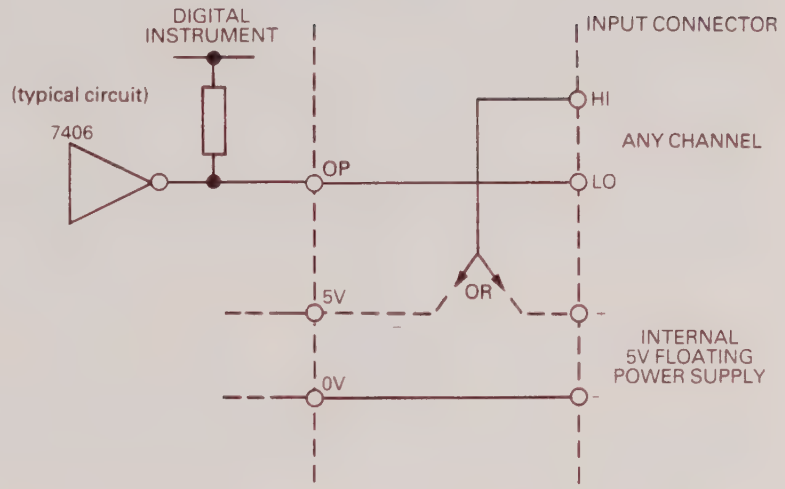
In this example, switch closure gives a measurement of logic 0.

(b) Up to 20 channels

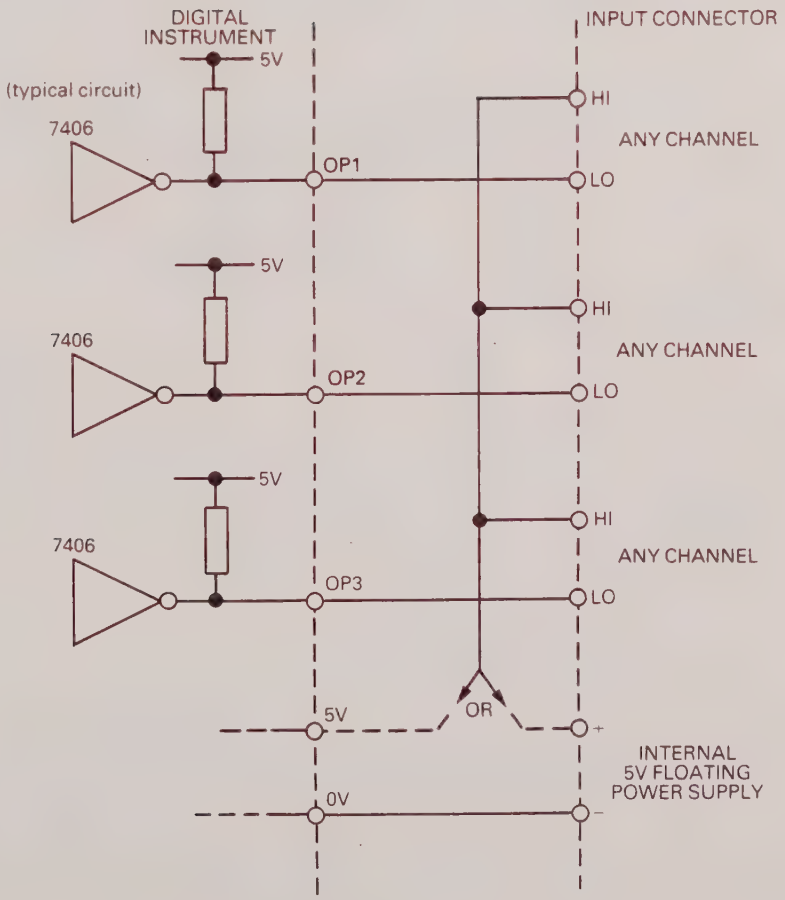


4.3 Outputs from digital instrumentation

(a) Single channel



(b) Up to 20 channels



5. When all the connections to the Input Connector have been made, loop the leads neatly around the cable clamp and through the cable tie, see Fig 5.2. Do not pass connecting leads over the top of the terminal blocks as this will prevent the cover plate from fitting properly, and will cause the leads to be trapped between the sharp edges on the blocks and the cover.

6. If desired enter any input information on the 'write-on' patch on the cover plate label. Replace the cover plate on Input connector.

7. Set links LK1 to LK20 on Event-Status Input Module (if necessary).
Note: Each input channel has an associated link which must be set according to the input voltage of each channel; LK1 corresponds with channel 1 (or the first of the group of 20 channels), LK2 corresponds with channel 2, etc. Set links as follows:

   input voltage 5V to 12V, link closed
   input voltage 12V to 80V, link open.

   Access to the links is gained by removal of the Event-Status Input Module. The procedure for removal of Module, connection of links and replacement of Module is given below.

   7.1 Switch off the logger

   7.2 Locate the extracting tool (supplied in the accessories bag) in the hole near front edge of Module and withdraw the Module.

   7.3 Using a soldering iron, connect or disconnect each of the links 1-20 (corresponding with channels 1 to 20, etc) as required; see Fig 5.5 for location of links.

   7.4 Replace Module in logger. Do not use extracting tool to replace Module.

8. Replace Input Connector.

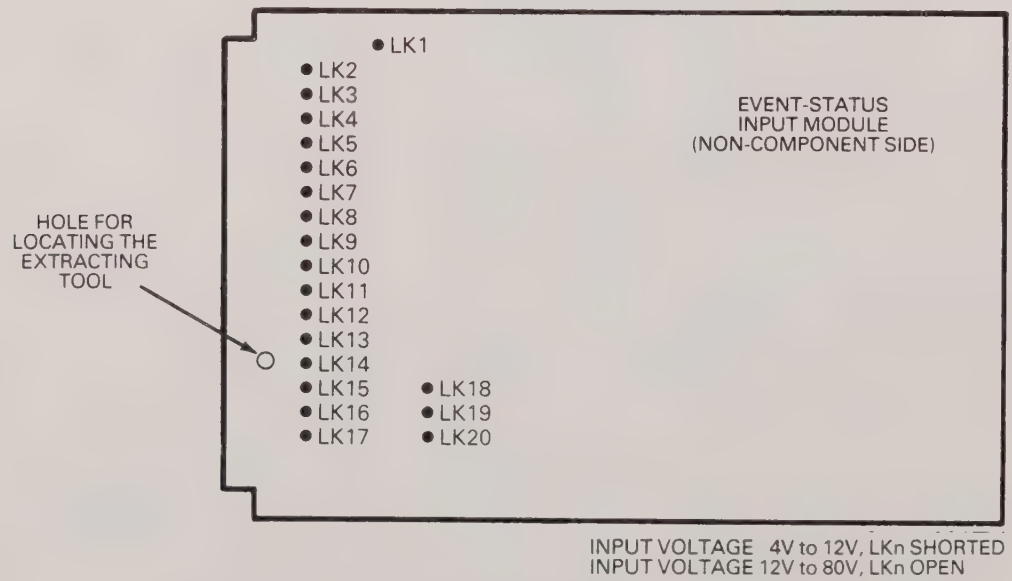


Fig 5.5 *Location of LK1-20 on Event Status Input Module*

2.2 FITTING ADDITIONAL CAPACITORS FOR GREATER CONTACT-BOUNCE FILTERING

Relay and switch contacts often suffer from 'contact bounce' when they are being closed. This effect is unavoidable, but the filtering circuitry on the Event-Status Input Module generally eliminates this effect. The input circuit for each channel contains a filter which provides immunity against contact-bounce up to approximately 800μs, this being adequate for the majority of applications. If, however, greater immunity is required, an additional capacitor can be fitted to the filter circuit of each or any channel. Each optional capacitor should be soldered to a respective pair of pins on the pcb. One pair of pins is provided for each input channel; their location is shown in Fig 5.6.

Capacitor values for three increased debounce times are shown below.

Capacitor value	*De-bounce time (approximate)*
10nF	1.6ms
22nF	2.5ms
33nF	3.4ms

The effect of additional capacitance is to increase the uncertainty in any event-time logged for that channel.

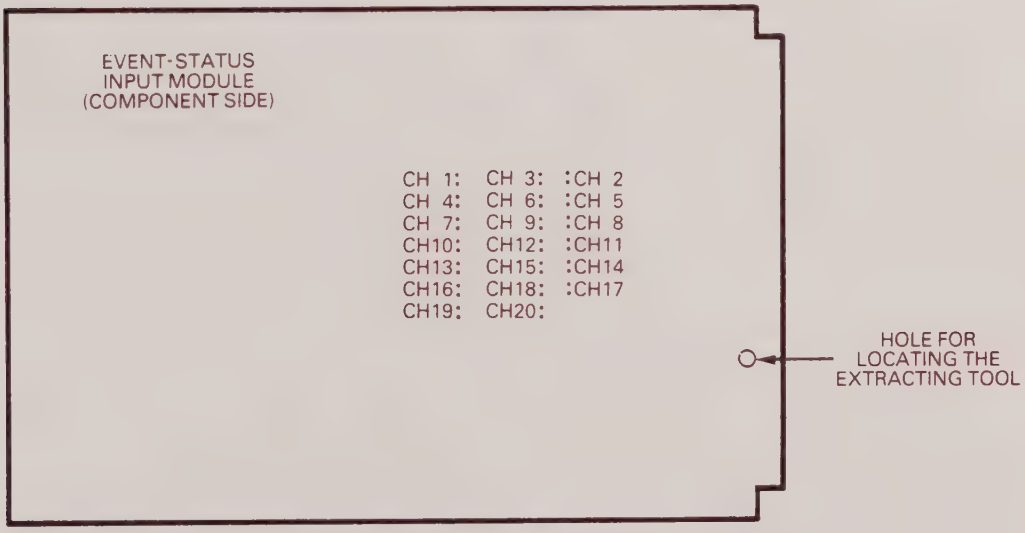


Fig 5.6 *Location of disconnect pins for optional filter capacitors*

Procedure

1. Switch logger off.
2. Remove the respective Input Connector.
3. Locate the extracting-tool (supplied in the accessories bag) in the hole near front edge of Module and withdraw the Module.
4. Solder capacitors to the pins provided, as required.
5. Replace Module and Input Connector; do not use extracting tool to replace Module.

Chapter 6
Using the Digital Input-Output Module (35302B)

Contents

		Page No
1	General Information	6.2
1.1	Introduction	6.2
1.2	Status Inputs	6.2
1.3	Alarm Outputs	6.3
1.4	Stimulus Outputs	6.3
1.5	External Measurement Modules	6.3
1.6	Setting up Alarms for Status Inputs	6.8
1.7	Channel Numbering and Pin Assignments	6.10
1.8	Avoiding multiple use of an Input or Output	6.11
2	Connecting the Inputs	6.12
2.1	General	6.12
2.2	Procedure	6.12
2.3	Specifications	6.12
3	Connecting the Outputs	6.13
3.1	General	6.13
3.2	Procedure	6.14
3.3	Specifications	6.16

1 GENERAL INFORMATION

1.1 INTRODUCTION

Each Digital Input/Output Module (35302B) provides the following digital input and output capability:

Inputs: 20 input lines, floating with common return
Outputs: 20 output lines, floating with common return

The common returns for the inputs and outputs are internally connected to the same point.

The inputs and outputs can be configured for use in one of two ways:

1. The inputs and outputs are used independently. Inputs provide status information, and the outputs give alarm or control (stimulus) information. Refer to the following sections for further information: '1.2 Status Inputs', '1.3 Alarm Outputs' and '1.4 Stimulus Outputs'.

2. The inputs and outputs are used in conjunction to read in data from other devices under supervision from the logger. Address and handshake arrangements are used to co-ordinate the inputs and outputs. Refer to '1.5 External Measurement Modules'.

The logic convention adopted for this unit conforms to that already described for the Status-Event Module, i.e.:

Logic 0 represents the presence of a current or voltage.
Logic 1 represents the absence of a current or voltage.

The actual voltage levels which the logger recognises as a 1 or a 0 are defined in the Specification.

1.2 STATUS INPUTS

The channels specified as status inputs are scanned at the times and intervals defined in a Task. In scanning the status inputs, the logger observes the input voltage level, eg 0V or 5V. The scanning speed is approximately 500 channels/s. Status inputs can be used as individual channels, or they can be grouped. The logic state (1 or 0) displayed for each separate input channel will be dependent on the input voltage.

if >3.5V	logic 0 displayed
if <1.5V	logic 1 displayed.

The alarm state of any individual channel can be defined as 1 or 0. Alternatively, a number of consecutive channels (maximum of 8) can be grouped, and a group state can be defined as an alarm. A typical group state would be 1101 for a 4-channel group.

When used as a group, the channels are normally consecutive. In this case the whole group is considered as one channel, having the number of the lowest channel in the group (e.g. the 8 channel group 41-48 would be referred to as Channel 41).

1.3 ALARM OUTPUTS

Alarm outputs can be controlled from alarm conditions detected on any analogue measurements, or status and event inputs. Two alarm output channels can be allocated to any input channel or group of channels; the quiescent state of both alarm outputs is logic 0. When a low alarm state is detected, the 'lo' alarm output defined for the input(s) will change to a logic 1 and remain in that state until low alarm is no longer detected. When a high alarm state is detected, the 'hi' alarm output defined will change to a logic 1 and remain in that state until a high alarm is no longer detected. If required, the same alarm output channel can be specified to give indication of both high and low alarm at the input(s). Alternatively, one alarm output only can be specified, in which case only high or low alarm would be indicated. Alarm outputs are reset when the logger is Halted.

Some alarm conditions are transient, i.e. 'Change' and 'Threshold'. The alarm output will pulse on (change to a logic 1 state (when the alarm occurs, and is reset (changes back to a logic 0 state) by the next (non-alarm) reading on the particular channel. Since this pulse at the output is very short, the user may wish to effectively prolong the pulse by connecting external latching circuitry to the alarm output.

1.4 STIMULUS OUTPUTS

Stimulus outputs are intended for controlling external circuitry, and for switching devices such as pumps or heaters. In the simplest concept, they can be regarded as similar to the alarm outputs. Stimulus outputs, however, offer greater flexibility. They may, for instance, be used in groups to produce changing output patterns which control a test rig through various phases of operation.

Stimulus outputs are controlled by the processing attached to a Scan Task. The outputs may be controlled individually or in groups. When used as a group the outputs form a bit pattern i.e. a combination of logic 1's and 0's. Stimulus outputs are reset when the logger is Halted.

1.5 EXTERNAL MODULE MEASUREMENT

The operation of the inputs and outputs can be combined to read binary or bcd data from external modules. Such modules could include remote counters, a-to-d converter packs or other instrumentation.

The input lines on PL1 can be used to read data sequentially from an external module. Up to 3 bytes of 8-bit binary data, or 6 bcd decades (excluding 'sign' decade), can be read. Normal bcd (negative logic) or inverted bcd (positive logic) can be input; this is selected as part of the Channel Definition, see Chapter 9. A handshake routine controls the data transfer, and a digitise command can be obtained from the logger if required. The remote modules may require a small amount of additional circuitry.

Using one 35302B, up to 20 external modules can be commanded to digitise simultaneously or sequentially; the data then being read into the logger.

The external modules and decoding circuitry used in a specific application will depend on user's requirements and cannot, therefore, be defined in this manual. An example configuration is however shown in Fig 6.1.

Input and Output organisation The organisation of inputs and outputs when using the External Module Measurement mode of operation is shown below. The inputs 1-10 and outputs 1-10 refer to those contained on PL1. See Fig 6.2 for pin numbering. (Fig 6.2 shows organisation of inputs/outputs when used as Status Inputs and Alarm and Stimulus Outputs).

Inputs

1 LSB
2
3
4
5 } BINARY/BCD DATA INPUTS
6
7
8 MSB
9
10 DIGITISE COMPLETE { DIGITISATION COMPLETE / DIGITISATION NOT COMPLETE } signal from Module indicating that digitisation is complete

NB The Digitise Complete signal must remain in the logic hi state (Digitisation Complete) until cleared by the next Digitise Command pulse, output at pin 9 (PL1 Outputs).

Outputs

1, 2, 3 SELECT NUMBER OF BYTES (1-3) or BCD DECADES (1-6) excluding sign decade

OUTPUT	BYTES/DECADES 1	2	3	4	5	6
1	1	0	1	0	1	0
2	0	1	1	0	0	1
3	0	0	0	1	1	1

4, 5, 6, 7, 8 MODULE ADDRESS OUTPUTS Selects one module, 1 to 20 (see Note 1 below)

OUTPUT	MODULE 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
4	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
5	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
6	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1
7	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

9 DIGITISE COMMAND — 20 µs — output after DIGITISE COMPLETE is received; intended to start digitisation in the Module.

10 ADDRESS VALID — ADDRESS VALID / ADDRESS NOT VALID — Valid when address outputs have settled.

Note 1 The Module Numbers correspond with the Channel Numbers; the actual 20-channel number-group depends on location of Digital Input/Output Module in mainframe. The Module Numbers are defined as 'Channel Numbers' in the Task Definition.

Simple Example A simple arrangement using one set of 10 inputs and 10 outputs is shown in Fig 6.1. In this example, the object is to read the digitised output from a 14 bit a-to-d converter. The 14 output lines are connected to two 8-line latches which can be selected independently; this permits the 14 outputs to be read in 2 binary bytes (8 lines and 6 lines).

Inputs 1-8 are regarded as one channel; in this example, Channel 1. The external module (the 14 bit a-to-d converter) is addressed by the Module Address output; the external address Decoder is enabled by the Address Valid output (at output 10) going to the logic 1 state. The Byte Output is set to byte 1, causing one of the external Latches to be enabled in readiness for the first data byte.

The Digitise Command signal at Output 9 is a 20 μs positive going pulse. This signal is produced when all the outputs (1-8) are stable, and it causes the Module to start reading. The Module digitises its analogue input (sample and hold). When the measurement is complete the Module outputs a positive going Digitise Complete signal which is input to the logger (input 10), this indicates to the logger that the data from the Module is ready to be read. The logger then reads in the first byte at inputs 1-8.

The Byte Select output now selects Byte 2 causing the other Latch to be enabled in readiness for the logger to read the second byte. When the outputs have settled the Address Valid output goes to logic 1. The logger then reads the second byte at inputs 1-6.

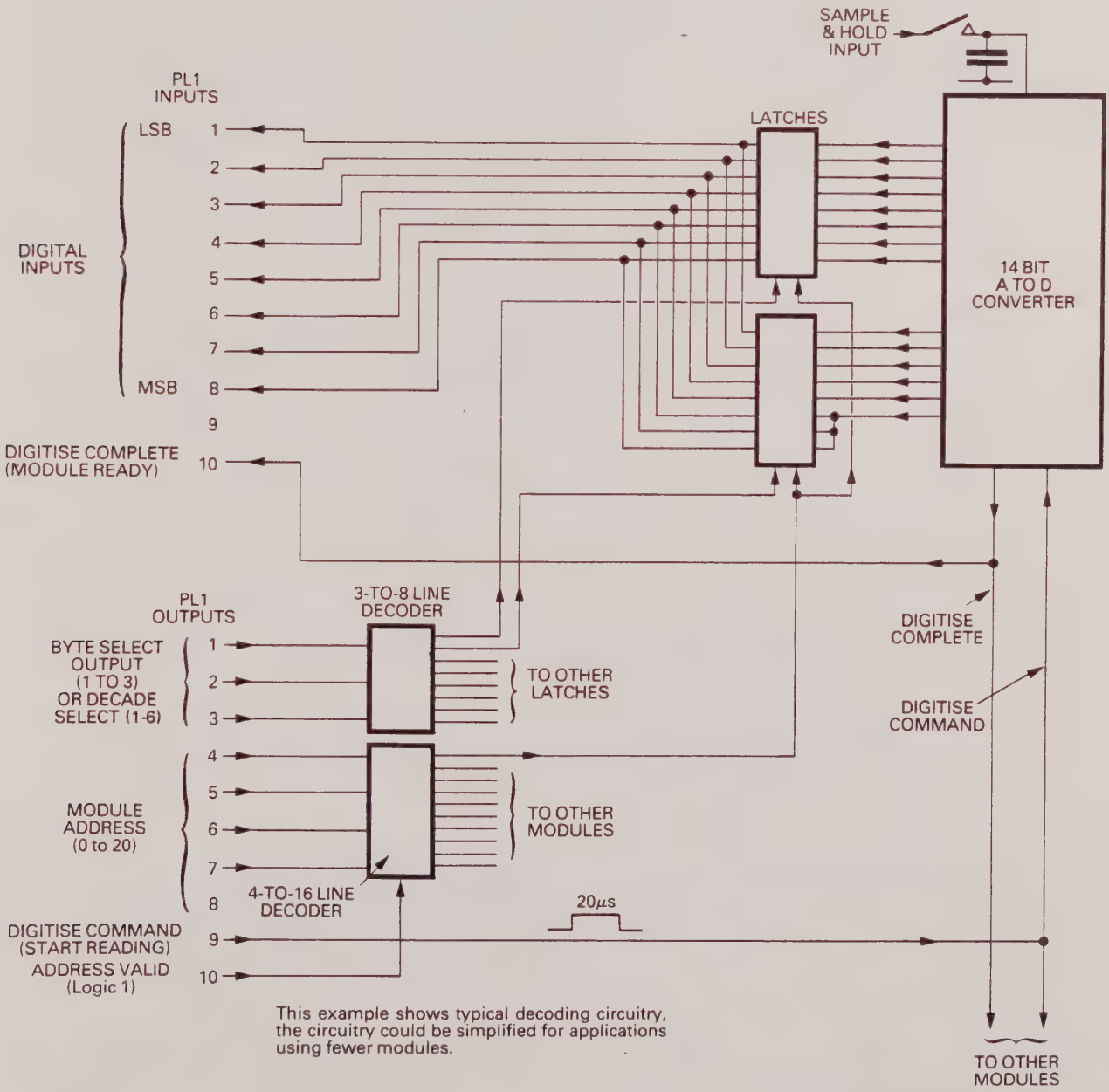
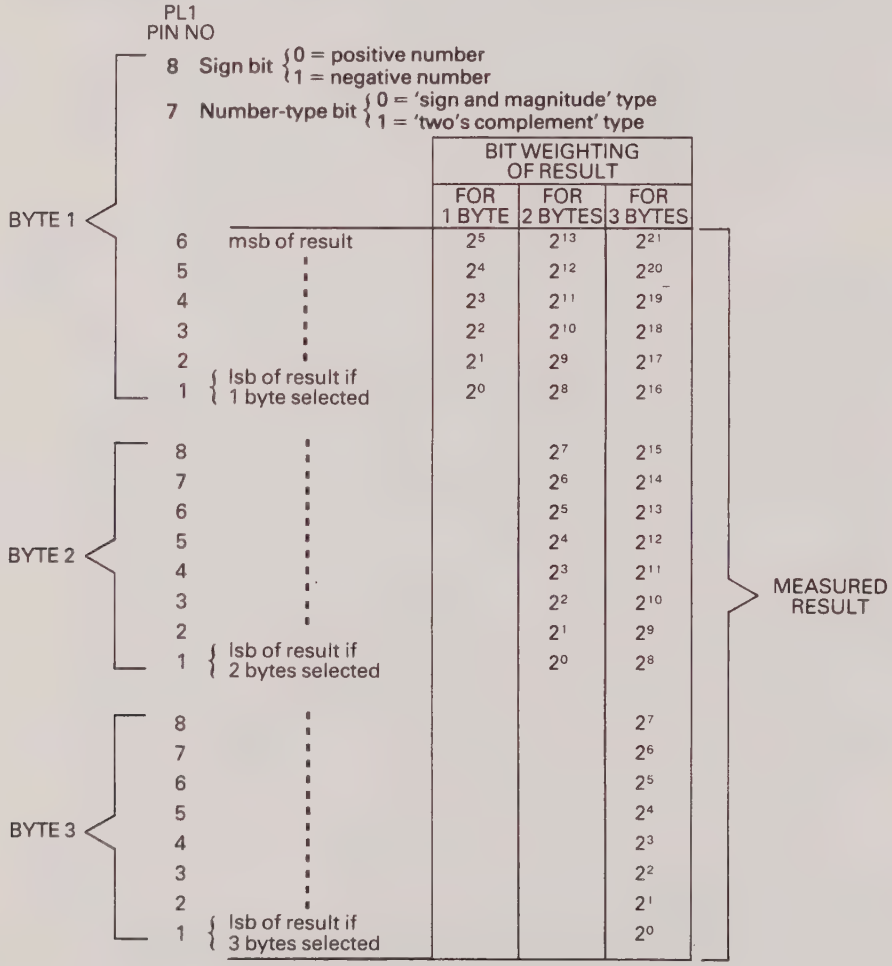


Fig 6.1 *Typical connections for External Measurement Modules*

Binary or bcd inputs may be connected to PL1. The significance of each individual input in each binary byte or bcd decade is shown below.

Binary Inputs


PL1 PIN NO
8 Sign bit 0 = positive number 1 = negative number
7 Number-type bit 0 = 'sign and magnitude' type 1 = 'two's complement' type
BIT WEIGHTING OF RESULT
FOR 1 BYTE
FOR 2 BYTES
FOR 3 BYTES
BYTE 1
6 msb of result
1 lsb of result if 1 byte selected
BYTE 2
1 lsb of result if 2 bytes selected
BYTE 3
1 lsb of result if 3 bytes selected
MEASURED RESULT

1, 2 or 3 bytes may be input for each measurement; the bit weighting in each byte varies with the number of bytes selected.

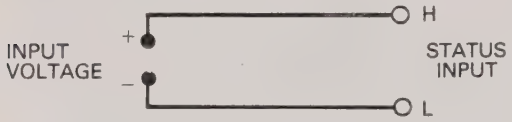
BCD inputs

		PL1 PIN NO		BIT WEIGHTING DECADES 1	2	3	4	5	6
BYTE 1	Sign decade	8	Sign bit 1 = − ve 0 = + ve						
		7	not used						
		6	not used						
		5	not used						
	1st decade	4	most significant decade	8	80	800	8000	80000	800000
		3		4	40	400	4000	40000	400000
		2		2	20	200	2000	20000	200000
		1		1	10	100	1000	10000	100000
BYTE 2	2nd decade	8			8	80	800	8000	80000
		7			4	40	400	4000	40000
		6			2	20	200	2000	20000
		5			1	10	100	1000	10000
	3rd decade	4				8	80	800	8000
		3				4	40	400	4000
		2				2	20	200	2000
		1				1	10	100	1000
BYTE 3	4th decade	8					8	80	800
		7					4	40	400
		6					2	20	200
		5					1	10	100
	5th decade	4						8	80
		3						4	40
		2						2	20
		1						1	10
BYTE 4	6th decade	8							8
		7							4
		6							2
		5							1
	7th decade	4							
		3							
		2	(Not used)						
		1							

Any number of bcd decades between 1 and 6, excluding the Sign decade, may be input for each measurement; the bit weighting in each decade varies with the number of decades selected. The decades are read in pairs, in up to 4 bytes.

1.6 SETTING UP ALARMS FOR STATUS INPUTS

(a) Single channels Consider the simple case of one status input channel with an input voltage applied.



INPUT VOLTAGE	MEASURED LOGIC STATE
> 3.5V	0
< 1.5V	1

If the input voltage applied between the H and L terminals is >3.5V then a logic 0 is detected, if the voltage is <1.5V then a logic 1 is detected. This establishes the input condition for a measured result of logic 1 or 0.

To set up an alarm condition for this channel the user must select Match or Mismatch, and a logic state.

Alarm on Match: Select 1 or 0, this represents an alarm level of logic 1 or 0 respectively. When the detected logic level is the same as the level selected (i.e. the levels Match) then an alarm will occur. No alarm occurs if the levels are different.

Alarm on Mismatch: Select 1 or 0, this represents an alarm level of logic 1 or 0 respectively. When the detected logic level is different from the level selected (i.e. the levels Mismatch) then an alarm will occur. No alarm occurs if the levels are the same.

The Match/Mismatch level is known as a 'pattern'; the reason for this will become evident after the reader has considered the alarm condition for grouped status channels, see below.

(b) Grouped channels As explained in a previous section, status inputs may be used individually or as a group of up to 8 channels. For simplicity, it is normal for any group to consist of consecutive channels, e.g. channels 21 to 28 would be used as an 8 channel group. The whole group is referred to by the lowest channel number only, i.e. 21 (the other numbers in the group are not used). When a group of channels is scanned, the output from each channel is detected and the results for the whole group is output as a pattern of logic 1's and 0's. For example, the 8 channel group referred to as channel 21 might produce the following result:

Ch 21 00110010

This corresponds with the separate channels in the group as follows:

Channel Number	28	27	26	25	24	23	22	21
Logic Level Detected	0	0	1	1	0	0	1	0

Note that the lowest numbered channel is always presented on the right, the highest on the left. The significance of all logic 1's or 0's detected is identical to that described under 'Single Channels' above.

To set up an alarm condition for grouped channels the user must select pattern Match or Mismatch, and a pattern of logic 1's and 0's (or ⊛ for 'dont care' channels, see below).

Alarm on Match: Select a logic 1 or 0 for each channel in the group, e.g. for an 8 channel group set up a total of 8 logic 1's or 0's; this could typically be: '00010001'. This represents an alarm pattern. When the measured pattern is the same as the alarm pattern selected (i.e. all 'bits' in the pattern Match) then an alarm will occur. No alarm occurs if the patterns are different.

e.g. 1	Measured pattern	01010000	
	Alarm pattern selected	00010001	= No alarm
e.g. 2	Measured pattern	00010001	
	Alarm pattern selected	00010001	= ALARM condition

Alarm on Mismatch: Select a logic 1 or 0 for each channel in the group, e.g. for an 8 channel group set up a total of 8 logic 1's and 0's, this could typically be '11100001'. This represents an alarm pattern. When the measured pattern is different from the alarm pattern selected (i.e. the patterns Mismatch) then an alarm will occur. No alarm occurs if the patterns are identical.

e.g. 1	Measured pattern	01010000	
	Alarm pattern selected	01010000	= No alarm
e.g. 2	Measured pattern	01000000	
	Alarm pattern selected	01010000	= ALARM condition

Note that an alarm pattern, for Match or Mismatch, is entered from the keypad as follows:

the bit corresponding with the highest numbered channel in the group is entered first, the bit corresponding with the lowest numbered channel in the group is entered last.

Thus if alarm pattern 00001111 is entered for channels 21 to 28 the bits correspond with the channels as follows:

Displayed pattern	0	0	0	0	1	1	1	1
Corresponding channel numbers	28	27	26	25	24	23	22	21

Dont-care This is a masking facility which allows the user to make any of the bits in an alarm pattern into a 'dont-care' state. When any bit is defined as 'dont-care' the detected level (Logic 1 or 0) of the corresponding channel will have no effect on the alarm state, but the level detected will be output at the output device in the normal way. To set up a dont-care state, simply enter ⊛ instead of a 1 or 0. For example, instead of the alarm pattern shown above (00001111) for channels 21 to 28, the user could enter 0000⊛111. This would mean that whatever level (Logic 1 or 0) was detected at channel 24, it would have no effect on the alarm state. The detected level of all channels, including channel 24, would however be output to the output device in the normal way when the channels were scanned.

Alarm outputs for Match/Mismatch For each single status channel or each group of status channels, only one alarm channel should be specified for Match/Mismatch alarm indication. The alarm output channel will change from logic 0 to logic 1 when a Match or Mismatch is detected.

1.7 CHANNEL NUMBERING AND PIN ASSIGNMENTS

The inputs and outputs are connected to the sub-miniature D-type connectors PL1 and PL2 located on the Digital Input/Output Module. The 20 inputs and 20 outputs are grouped as 2 sets of 10 inputs and 10 outputs, see Fig 6.2.

Outputs 1-10 and Inputs 1-10 are located on PL1; Outputs 11-20 and Inputs 11-20 are located on PL2. Note that, although the inputs and outputs are referred to above and in Fig 6.2 as '1-20', the actual numbering will depend on the location of the Digital Input/Output Module in the logger mainframe, see Fig 6.3.

DIGITAL INPUT/OUTPUT MODULE

PL1

PL2

PL1	
PIN NO	SIGNAL
1	Vu
2	0Vf
3	O1
4	O2
5	O3
6	O4
7	O5
8	O6
9	O7
10	O8
11	O9
12	O10
13	
14	0Vf
15	I1
16	I2
17	I3
18	I4
19	I5
20	I6
21	I7
22	I8
23	I9
24	I10
25	

OUTPUTS

INPUTS

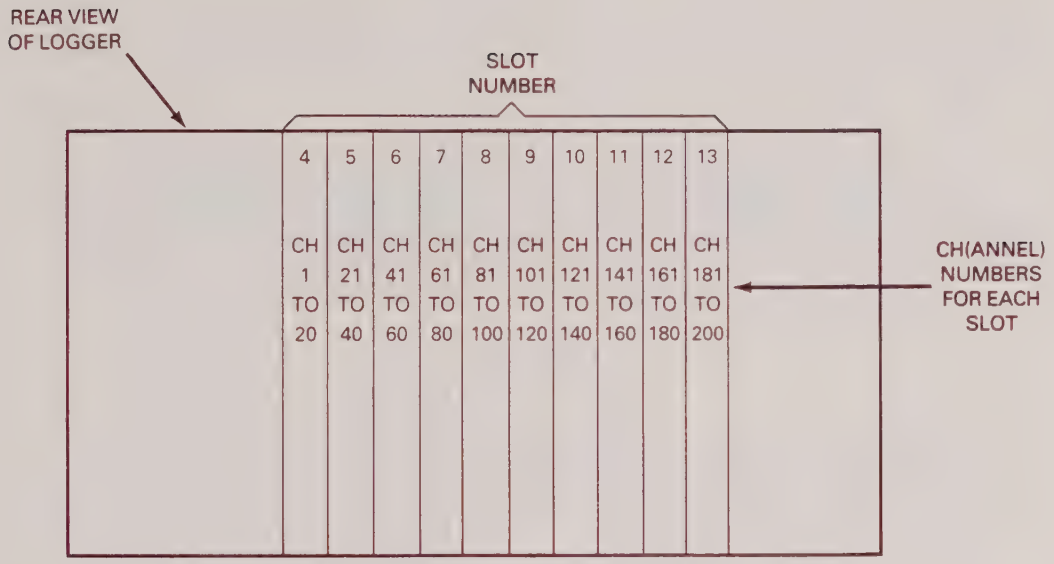
PL2	
PIN NO	SIGNAL
1	Vu
2	0Vf
3	O11
4	O12
5	O13
6	O14
7	O15
8	O16
9	O17
10	O18
11	O19
12	O20
13	
14	0Vf
15	I11
16	I12
17	I13
18	I14
19	I15
20	I16
21	I17
22	I18
23	I19
24	I20
25	

OUTPUTS

INPUTS

Fig 6.2 *PL1/PL2 Pin assignment*

The inputs and outputs, as shown in Fig 6.2, are valid for Status Inputs, Alarm Outputs and Stimulus Outputs. *Note that some of the inputs and outputs have a different significance when used in the 'External Measurement Module' mode, see '1.5 External Measurement Modules'.*



An Input Connector and the associated Reed Relay or Solid State Selector may be fitted in any of the slots 4-13 in the logger mainframe. The Connectors and the Selectors are interchangeable; channel numbering is however always dependent on position in the mainframe.

Fig 6.3 *Simplified rear view of logger showing channel numbering*

1.8 AVOIDING MULTIPLE USE OF ANY INPUT OR OUTPUT

As explained in the preceeding sections of this chapter, the Digital Input/Output Module will accommodate the following types of input/output.

STATUS inputs and/or ALARM outputs and/or STIMULUS outputs

or

EXTERNAL MODULE MEASUREMENT, the inputs/outputs at PL1 are wholly utilised in this mode of operation.

When programming the logger careful consideration should be given to the required use of all inputs and outputs, and multiple use must be avoided. For instance, it is permissible to connect STATUS inputs to the input channels and a mixture of ALARM and STIMULUS outputs to the output channels, but is not permissible to connect any of these inputs or outputs to PL1 if it is being used for EXTERNAL MODULE MEASUREMENT.

2 CONNECTING THE INPUTS

2.1 GENERAL

The status inputs are activated by a logic 1 or logic 0 input voltage. This can be achieved using a switch and an external supply voltage (there is no energisation from the logger for the status inputs). TTL-compatible semi-conductor switches or relay contacts may be used to switch the supply. The connection details below apply equally to STATUS inputs and the inputs in EXTERNAL MODULE MEASUREMENT.

2.2 PROCEDURE

1. Connect the inputs as shown in Fig 6.4; make connections to a sub-miniature D-type connector which will mate with PL1/PL2. Ensure that the input conforms to the specifications given in '2.3 SPECIFICATIONS' below.

2. Refer to the preceeding section in this chapter for general information on the permitted use of the inputs and channel numbering etc.

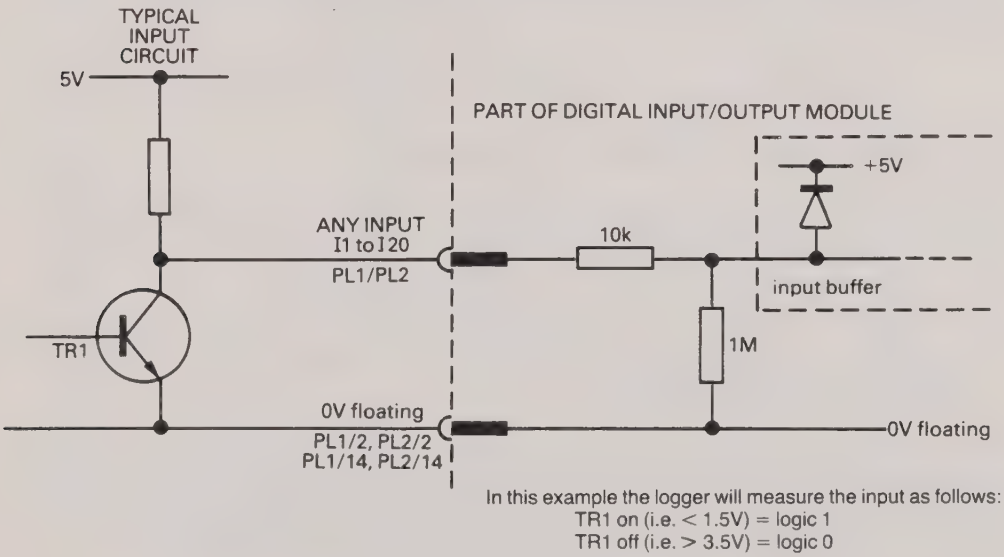


Fig 6.4 *Typical input configuration*

2.3 SPECIFICATIONS

Number of lines:	20, with common return
Input logic 1:	<1.5V
Input logic 0:	>3.5 V
Maximum input voltage:	15V
Maximum permitted common mode voltage:	100V with respect to logger earth (Note that the floating 0V return is common to digital inputs and outputs)

3 CONNECTING THE OUTPUTS

3.1 GENERAL

The digital outputs can be configured for one of three different types of electrical arrangement, regardless of whether the outputs are dedicated to ALARM, STIMULUS or EXTERNAL MODULE MEASUREMENT. The three configurations are:

(a) TTL/CMOS compatible outputs, using internal floating 5V supply.

(b) Outputs with 4.7k 'pull-up' resistors to external supply voltage.

(c) Open-drain (fet) outputs using an external supply.

Selection of (a), (b) or (c) above, which commits all 20 outputs to the same configuration, is achieved by appropriate connection of links P1-P5 on the Module. The location of these links is shown in Fig 6.5.

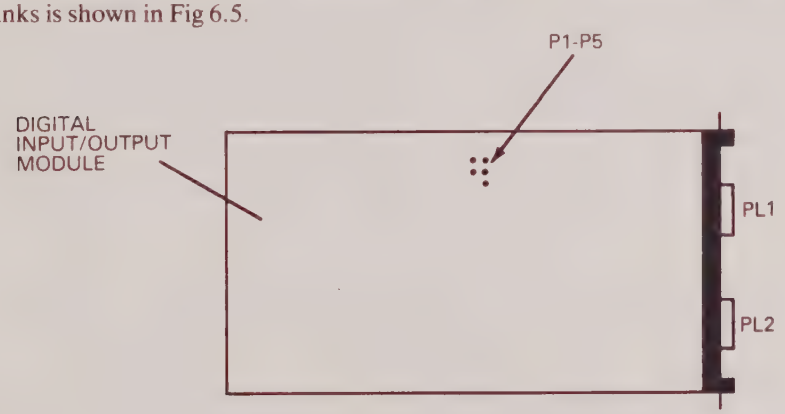


Fig 6.5 *Location of pins P1-P5*

The linking required to achieve the desired configuration is shown in the wiring diagrams below, Figs 6.6 – 6.8. Each Digital Input/Output Module is supplied already linked for 'TTL compatible outputs'. In order to change the linking of P1-5, switch off logger and remove the Module.

3.2 PROCEDURE

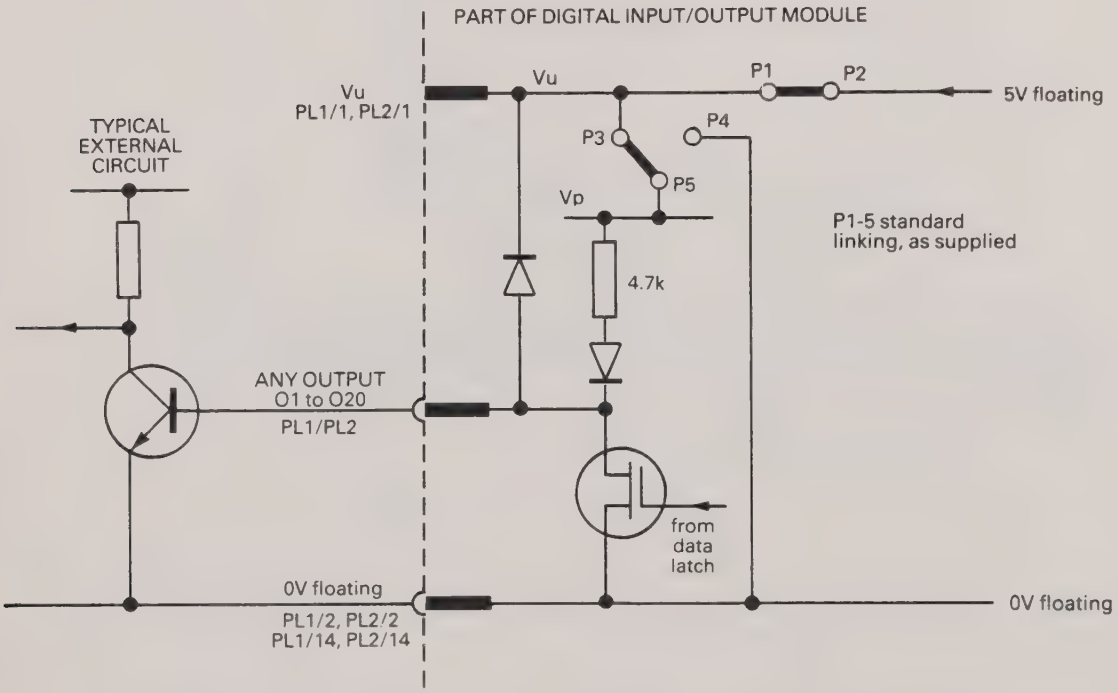
1. Switch off logger. Remove Digital Input/Output Module (rotate ¼-turn locking bolts and withdraw Module) and note the setting of links P1-P5, see Fig 6.5. Ensure that the links are set correctly for the type of output required:

   (a) TTL compatible outputs using internal supply, see Fig 6.6.
   or
   (b) 4.7k 'pull-up' resistor outputs using external supply, see Fig 6.7.
   or
   (c) Open-drain fet outputs using external supply, see Fig 6.8

2. Replace the Digital Input/Output Module, lock in position with the ¼-turn locking bolts.

3. Refer to the preceeding sections in this Chapter for general information on the permitted use of outputs, and channel numbering.

4. Connect all outputs as shown in Fig 6.6 or 6.7 or 6.8 as appropriate, corresponding with the type of linking (P1-5) selected in instruction 1. Make connections with sub-miniature D-type plug supplied, this mates with PL1/PL2; see Fig 6.2 for pin details. Ensure that the output conforms to the specifications given in '3.3 SPECIFICATIONS' below.



The selected logic state causes the internal fet to be switched as follows:
logic 1 = fet on
logic 0 = fet off

Fig 6.6 *TTL compatible outputs, using internal floating 5V supply*

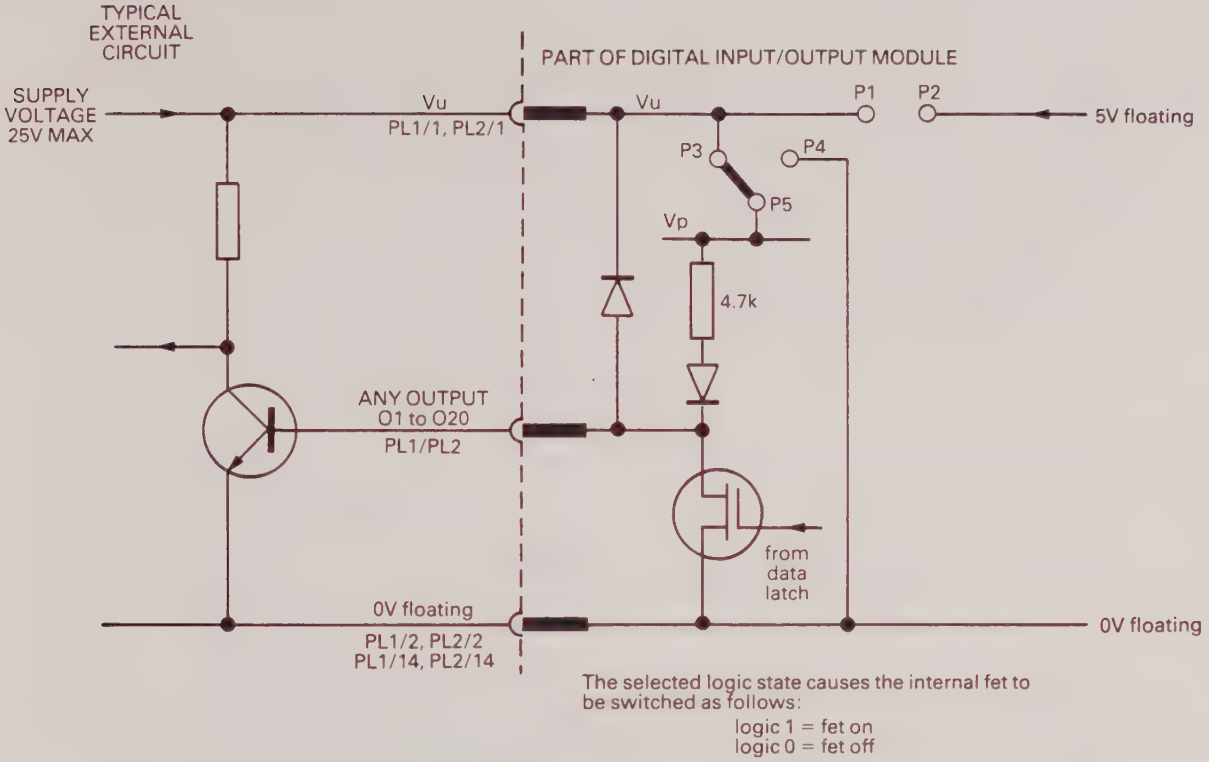


Fig 6.7 *Outputs with 4.7k pull-up resistors to supply voltage, using an external supply*

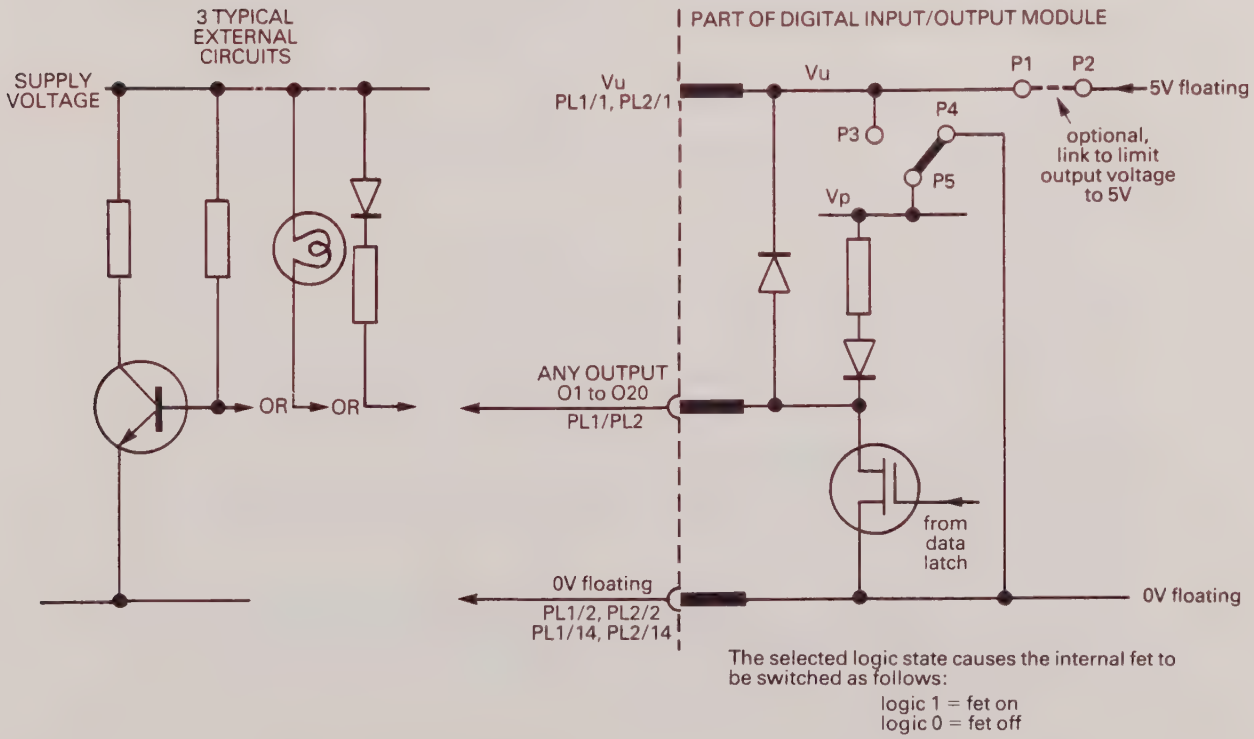


Fig 6.8 *Open-drain (fet) outputs using an external supply*

3.3 SPECIFICATIONS

Specifications common to all 3 configurations.

Number of lines:	20, with common return
Output switches (20):	Vmos- type fet
Output configurations (selected by links P1-P5):	TTL compatible (see Fig 6.6), Resistor pull-up to user supply (see Fig 6.7) Open drain fet (see Fig 6.8)
Maximum sink current for output voltage <0.4V (= logic 1):	20mA
Maximum sink current for typical output voltage of 1V:	180mA
Maximum dissipation per fet switch (0-60°C):	500mW
Maximum current per fet switch:	250mA
Maximum total common current:	1A per connector (PL1 or PL2) 2A total per module 0.5A per 0V return (PL1/2, PL2/2, PL1/14 PL2/14)
Maximum permitted common mode voltage:	100V with respect to logger earth (Note that the floating 0V return is common to digital inputs and outputs)
Output leakage current, logic 0 (=fet off):	<10μA
Fail-safe condition:	fet off (Logic 0)

TTL compatible outputs only (Fig 6.6)

Logic 0 (at 0mA):	>4V

Outputs with 4.7k pull-up resistor to supply voltage only (Fig 6.7)

Maximum (external) supply voltage:	25V

Open drain fet outputs only (Fig 6.8)

Maximum (external) supply voltage:	40V
Output voltage clamping:	Output clamped to any voltage by connecting Vu terminal to the clamp voltage supply.

Chapter 7
Using the Counter Timer Module 35302C

Contents

		Page No
1.	Introduction	7.1
2.	The Measurement Functions	7.2
2.1	Frequency	7.2
2.2	Period	7.5
2.3	Totalise	7.7
2.4	Interval	7.8
3.	Obtaining Measurements	7.9
4.	Using the F and G Inputs	7.10
5.	Input Connections and Switch Settings	7.11
6.	Transducer Energisation	7.17
7.	General Procedure for Making Connections and Switch Selection	7.18

1 INTRODUCTION

The Counter Timer Module 35302C permits any of the following four measurement functions to be selected:

Frequency
Period
Totalise
Interval

The Module has five input channels, any one of which can be used for any of the measurement functions.

The preferred location for a Counter Timer Module in the ORION frame is indicated on page 1.13. Note that the combined total of Counter Timer Modules and Analogue Output Modules must not exceed five per frame.

NB The Counter Timer Module can only be used with an ORION which has a modification status of 'B3' or higher. The modification status is indicated on the Serial Number Label which is located under the right-hand carrying handle (as viewed from front of instrument). If the status is lower than B3, consult a Solartron Service Department.

2 THE MEASUREMENT FUNCTIONS

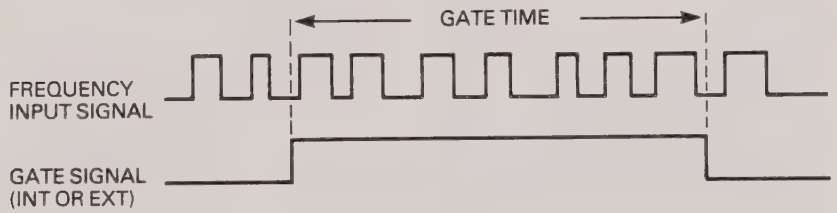
Each of the four Counter Timer measurement functions is described on the following pages.

2.1 FREQUENCY

Each result is the average measured over a selected 'gate' time. The gating can be selected as internal or external.

Internal Gate: One of four gate times may be selected: 0.1, 1, 10 or 100 seconds.

External Gate: External gating pulses, of any duration up to 600 seconds, may be input at the G terminal.

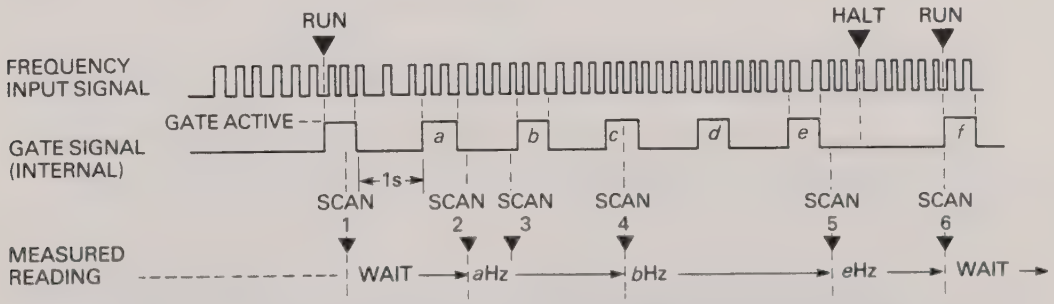


RESULT = AVERAGE FREQUENCY MEASURED OVER THE SELECTED GATE TIME

Frequency inputs up to 400 kHz may be measured. When measuring high frequencies, a short gate time may be used, but when measuring low frequencies, or to obtain maximum measurement accuracy with any input frequency, a long gate time should be used.

Assuming the logger is in the RUN state, measurement activity will start when a gate signal is applied, but the result will not be output until a Scan occurs; see the diagram below.

With the logger in RUN, the Monitor facility can be used to obtain each result as it becomes available. If the logger is in HALT, Monitor will cause measurements to start when a gate is applied, as for RUN.



The diagram illustrates a frequency input signal, and a gating signal which is generated internally (this could be 0.1s, 1s, 10s or 100s). Following a RUN command, the gating pulses will be produced automatically; the gate pulses will be repetitive with a 1 second delay between each.

During a scan the most recent measurement is read. This will be the average frequency measured over the last complete gate time.

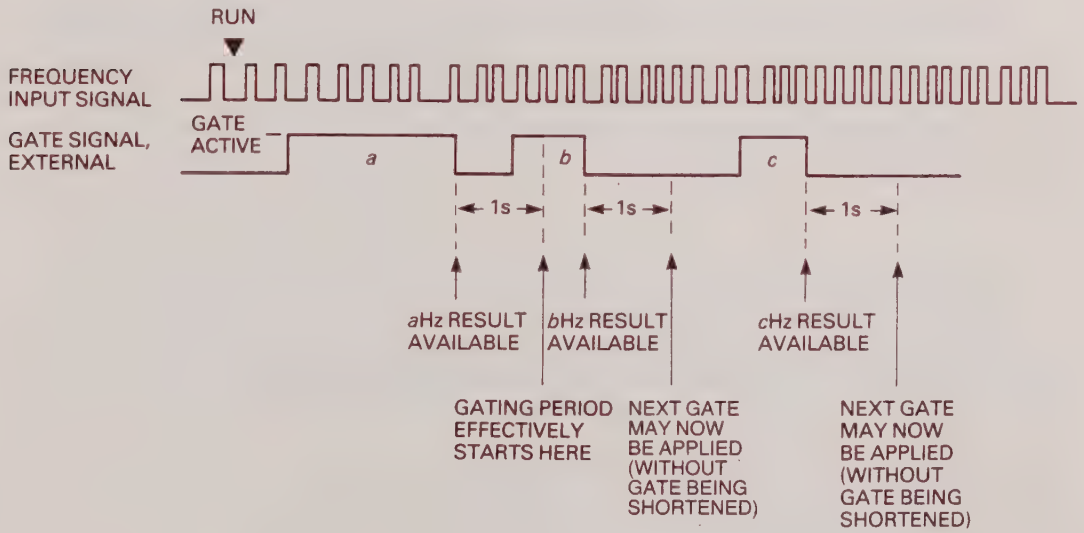
If the first gate time has not been completed before a scan occurs, the message 'Wait' will be output. The six typical scans shown in the diagram will produce the following results:

Scan 1	'Wait', no result ready yet
Scan 2	*a* Hz, as measured during gate time *a*
Scan 3	*a* Hz, since no further gate time has occurred
Scan 4	*b* Hz, as measured during gate time *b*
Scan 5	*e* Hz, as this was the latest measurement available
Scan 6	'Wait', since the RUN command effectively resets the frequency and no new gate time has yet been completed

At HALT, the gate pulses stop. At RUN, the result is reset and measurement re-starts.

Once a frequency measurement has been completed, a fresh measurement will not be started until one second has elapsed.

External gating will produce measured results in the same way as internal gating, and each new measurement will not be started until 1 second has elapsed. For example, if an external gate signal is applied for a certain time, and then re-applied before 1 second has elapsed, that part of the second gate signal which occurs during the 1 second interval will be ignored. This second gate signal will produce a normal frequency measurement, but measurement will not start until immediately after the 1 second interval, as shown in the diagram below.

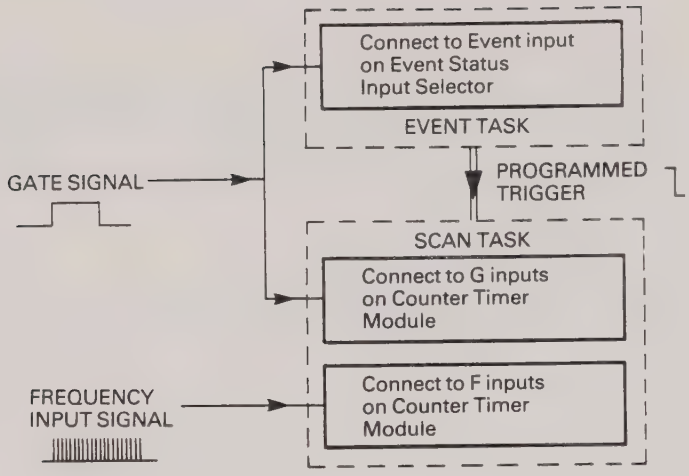


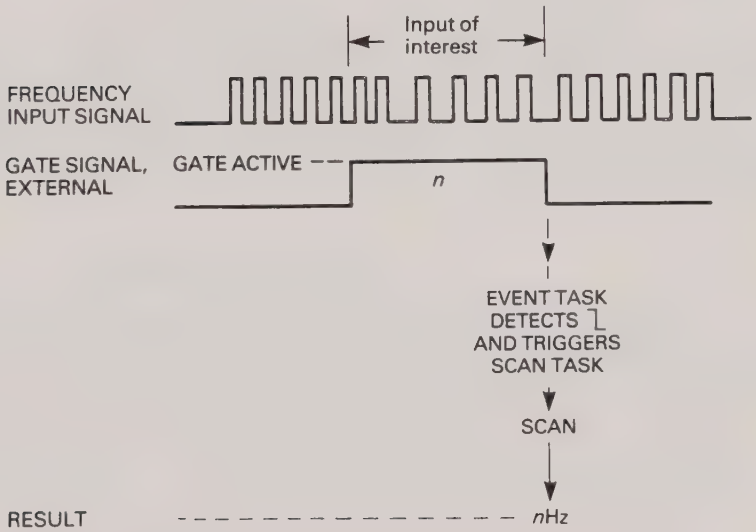
Thus, when using external gating, to avoid the gate time being shortened, allow a gap of at least one second between each gating pulse.

Furthermore, when external gating is employed, the user should ensure that each gate pulse occurs, and is completed, before a scan. Synchronisation between the end of a gate pulse and a scan may be obtained using the following method, which is illustrated overleaf.

1. Connect the gating signal to the G+ and G− inputs on the Counter Timer Module, and to an Event channel on an Event-Status Input Module.

2. Incorporate the Event channel in an Event Task, and program this Task to trigger the Scanning Task which is used to measure frequency.

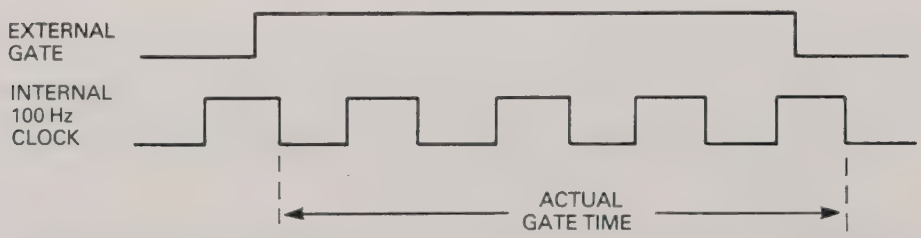

Connect to Event input on Event Status Input Selector
EVENT TASK
GATE SIGNAL
PROGRAMMED TRIGGER
SCAN TASK
Connect to G inputs on Counter Timer Module
FREQUENCY INPUT SIGNAL
Connect to F inputs on Counter Timer Module



Note that an external gate actually operates in conjunction with an internal clock signal which runs at 100 Hz. Thus, when the external gate is used, the actual gate time will be rounded up to the next internal clock edge. As a result of this, the actual gate may:

1. open later than the external gate by up to 10 ms
2. last for longer than the external gate by up to 10 ms

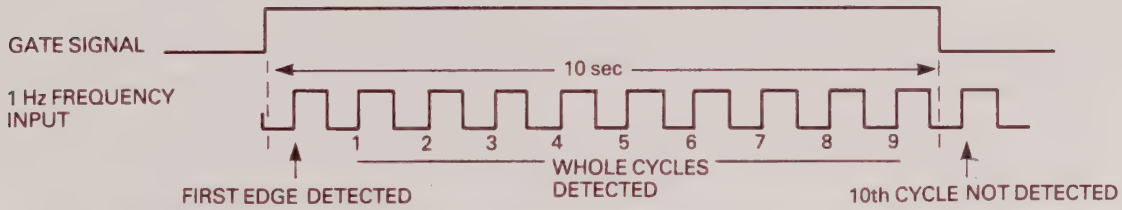
(This effectively limits the external gate time to greater than 10 ms)



This does not affect measurement accuracy, as the actual gate time is always used in the frequency calculation.

The Frequency measurement function is suitable for all but very low input frequencies. If the Frequency function is used to measure a frequency which is very low with respect to the gate time used, then a large measurement error may well be obtained. This is illustrated in the example below.

Consider a 1 Hz frequency being measured using a 10 second gate time.



In an ideal situation, the first edge of the frequency signal would coincide exactly with the start of the gating signal. In this case a precise measurement would be obtained. However, in a practical situation (as shown in the diagram above) these signal edges will not occur simultaneously. Since the first edge of the input signal may be up to 1 cycle time after the start of the gate, only 9 complete cycles can be measured, and a measurement error of up to 10% could occur.

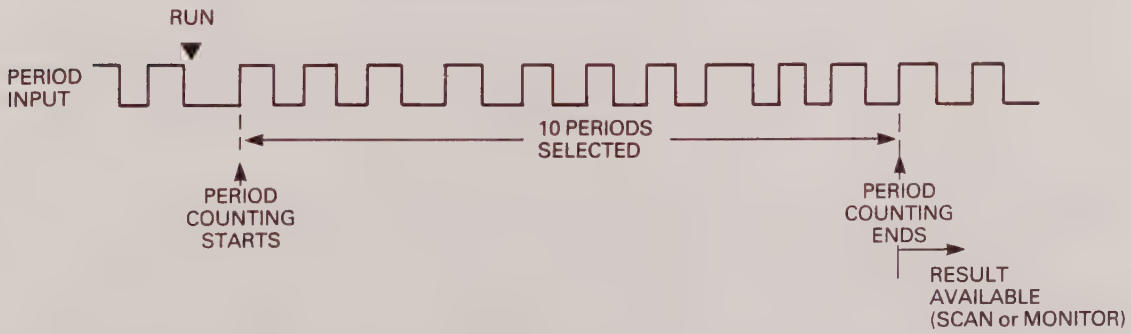
In order to produce an accurate measurement of a low frequency, the Period facility should be used instead of Frequency. The Period (time) result obtained can then be converted to frequency using the formula:

$$\text{Frequency (Hz)} = \frac{1}{\text{period (secs)}}$$

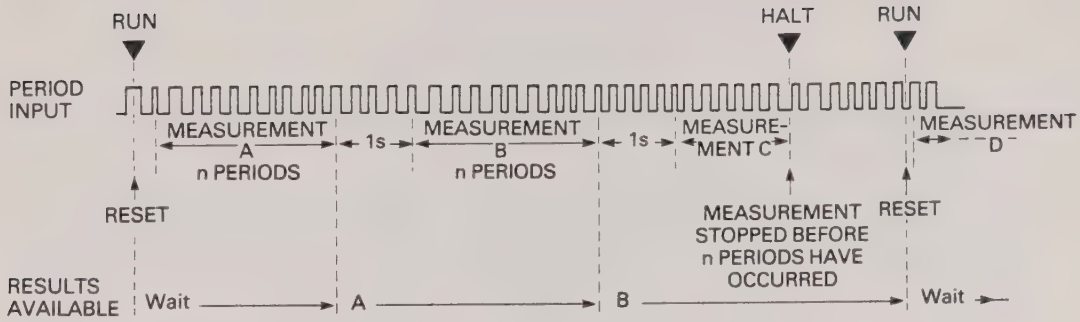
A Conversion may be used to achieve this automatically. The result thus obtained will be a highly accurate measurement of the low frequency. Period measurement is described below.

2.2 PERIOD

Each result is the average period, determined over a defined number of periods. 1, 10, 100, 1000 or 10,000 periods may be selected. (1 period = 1 complete cycle). The logger counts the selected number of input periods, measures the time elapsed, then divides the time by the number of periods, thereby producing the average period. Obviously, the highest accuracy is obtained by making the measurement over the highest number of periods.



The diagram above illustrates an example period input, for which '10 Periods' has been selected. The time taken for 10 periods to occur is measured, and will then be divided by 10. The result will then be available, and may be obtained by scanning or by monitoring. Measurement will be repetitive, with a 1 second interval between measurements, as shown overleaf.

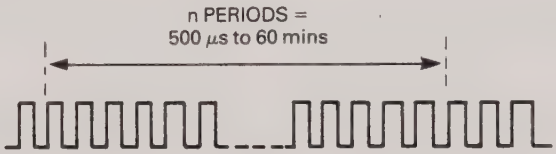


The example above again illustrates the number of periods selected (n) equal to 10, but n may be any of the selectable values. Note that each result becomes available at the end of each respective measurement; this result will be retained until the next measurement has finished. If a measurement is halted before n periods have occurred, no result will be obtained from the measurement. Pressing RUN will reset any existing result and restart measurement.

Once a period measurement has been completed, a fresh measurement will not be started until one second has elapsed.

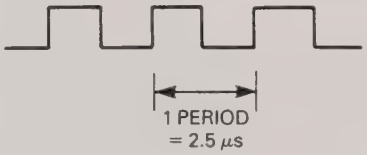
Period length which may be measured is defined as follows:

> the time taken to count n periods (1, 10, 100, 1000 or 10,000) must be in the range 500 μs to 60 minutes.



The Period measurement function should not be used to measure very short periods over a low number of periods, since poor accuracy may be obtained. This is illustrated in the example below.

Consider an input period of 2.5 μs (= 400 kHz) measured over 1 period. The logger can measure a period to an accuracy of approximately 1 μs. Since this represents a very large proportion of the time to be measured, a large measurement error can be expected.



Measurement accuracy can be improved in one of two ways:

1. the signal can be measured over a large number of periods (i.e. 10,000) or, preferably,
2. the Frequency facility should be used instead of Period.

The Frequency (Hz) result obtained can then be converted to period using the formula:

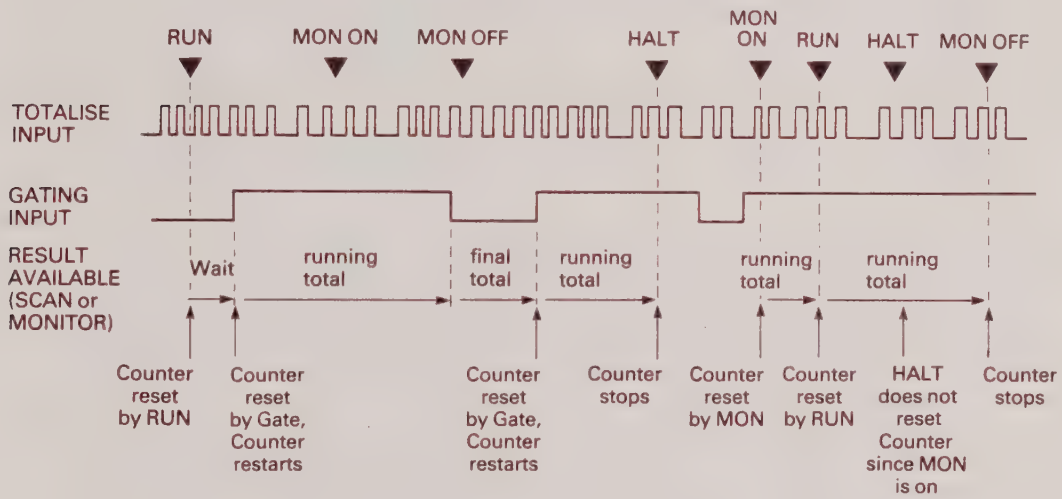
$$\text{Period (sec)} = \frac{1}{\text{Frequency (Hz)}}$$

A Conversion may be used to achieve this automatically. The result thus obtained will be a highly accurate measurement of the short period. Frequency measurement is described on pages 7.2 to 7.5.

2.3 TOTALISE

This produces a count of pulses which occurred while the gate was enabled. This function operates in the following manner:

Following a RUN command, the counter is reset to zero and, when a gating signal is applied, counting commences. When the gating signal is removed counting stops and the result is frozen. If the gating signal is again applied the counter is reset and counting re-commences.



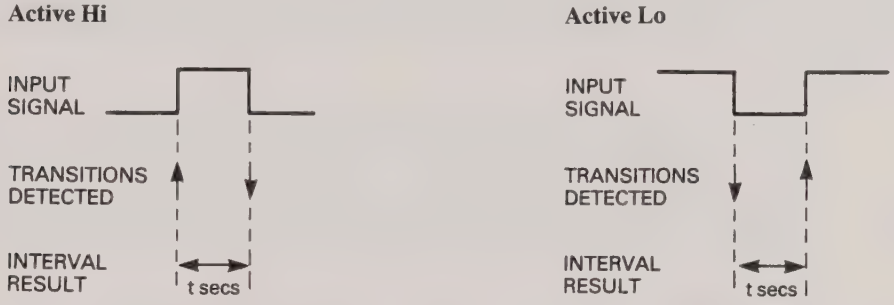
When a scan occurs, the running total or final total will be displayed. The counter will be reset whenever RUN is selected.

The Monitor facility may also be used to give a Totalise result. If the logger is in RUN, Monitor will produce the running total or final total. If the logger is in HALT, Monitor will reset the counter; counting will restart, assuming the gate signal is present, and the running total will then be shown.

It is preferable that the gate inputs are switched using electronic switches. If electro-mechnical switches are used they must be made bounce-free, possibly by the inclusion of a filter capacitor, since a make or break with a multiple bounce may cause the count to be erroneously reset.

2.4 INTERVAL

An interval is measured between a positive transition and the next negative transition (using Active Hi input wiring), or between a negative transition and the next positive transition (using Active Lo input wiring).



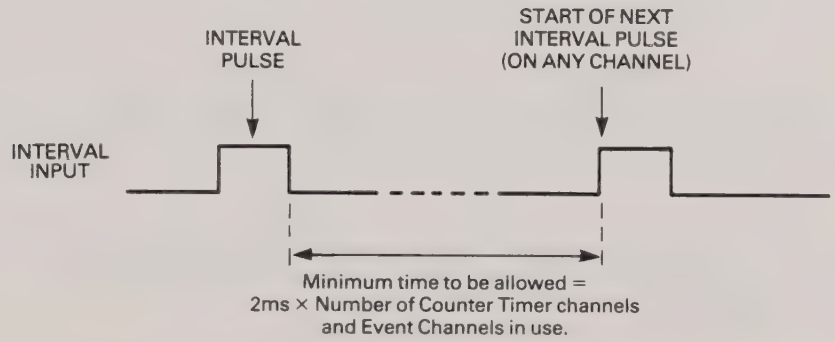
Internal times (t) within the range 500 μs to 60 minutes can be measured.

After each interval pulse, a minimum time must be allowed before another interval pulse can be applied to any channel; this includes all active Counter Timer channels, and all active Event channels on an Event Status Selector. The time to be allowed between interval pulses can be defined simply as:

$$\text{Time} = 2 \text{ ms} \times n$$

where n = total number of Counter Timer channels and Event channels within the logger which are in use.

For example, for 5 channels, allow 10 ms.


INTERVAL PULSE
START OF NEXT INTERVAL PULSE (ON ANY CHANNEL)
INTERVAL INPUT
Minimum time to be allowed = 2ms × Number of Counter Timer channels and Event Channels in use.

This time between interval pulses permits the processor to service any interrupts which any active channels may generate. If adequate time is not allowed between any two interval pulses, the result from the first pulse will be lost.

3 OBTAINING MEASUREMENTS

Counter Timer channels are active only when the logger is in the Run state, or when the Monitor is on. Results from Counter Timer channels may be obtained by scanning, with the logger in Run, or by monitoring, with the logger in Run or Halt.

Scanned results are produced in the normal way, by setting up a Channel Definition for each Counter Timer channel, and then incorporating these channels into a Task using Task Definition. Monitored results are obtained using the MON key in the normal way.

It is important to note the two operating conditions which will cause the results from all Counter Timer channels to be reset:

1. RUN command.

2. Monitor On command, if logger is in Halt state.

Measurement restarts immediately after reset.

The Run state has a higher priority than Monitor, such that, if Run is selected while the Monitor is on, the channels will be reset and measurements will restart. However, switching on the Monitor while in Run will not reset the channels, and the latest results available will be displayed.

The diagram below illustrates when the Counter Timer channels are reset by the RUN command, and by switching on Monitor.

The table below shows the action of the Counter Timer channels following any of the commands RUN, HALT, Monitor On or Off, while the current machine status is RUN, Monitor On or Off, or HALT, Monitor On or Off.

		CURRENT MACHINE STATE			
		RUN		HALT	
		MON ON	MON OFF	MON ON	MON OFF
COMMAND	RUN	(Error 118)	(Error 118)	Measurements restart	Measurements restart
	HALT	Measurements continue	Measurements stop	Measurements continue	No effect (Measurements are already stopped)
	MONITOR ON	Measurements continue	Measurements continue	Measurements restart	Measurements restart
	MONITOR OFF	Measurements continue	Measurements continue	Measurements stop	No effect (Measurements are already stopped)

The measurement activity defined in the above table will, of course, be dependent on the gating signal being present (where applicable)

Summary of Monitor operation

If logger is in HALT:

When MONITOR is switched on, measurement starts subject to the presence of a gating signal, if applicable. The 'Wait' message will be output until the first measurement result is available. This result will be maintained until the next is available, and so on. Measurements will continue until MONITOR is switched off.

Should RUN be selected while Monitor is on, the result will be reset and measurement restarted. This will again produce the 'Wait' message until this latest measurement is available. Monitor results then appear as described above.

If logger is in RUN:

When MONITOR is switched on, the most recent result obtained will be output. This result will be updated each time a new result becomes available.

Measurements started by the RUN command are unaffected when MONITOR is switched on or off.

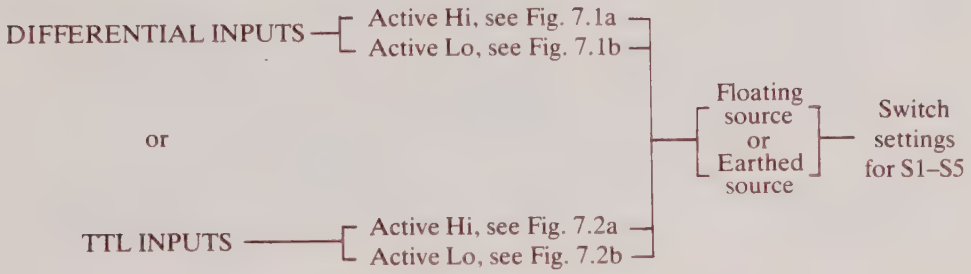
4 USING THE F AND G INPUTS

Each of the five channels consists of 2 pairs of inputs: F+ and F−, and G+ and G−. One pair, or both pairs, of inputs are used for each channel, depending on the measurement selected. The inputs to be used for each of the four measurement functions are indicated in the table below.

MEASUREMENT FUNCTION	F INPUTS	G INPUTS
Frequency	Frequency signal	External gating signal
Period	Period signal	Not used
Totalise	Totalise signal	Gating signal
Interval	Not used	Interval signal

Table 7.1 *Inputs to be used for each Measurement Function*

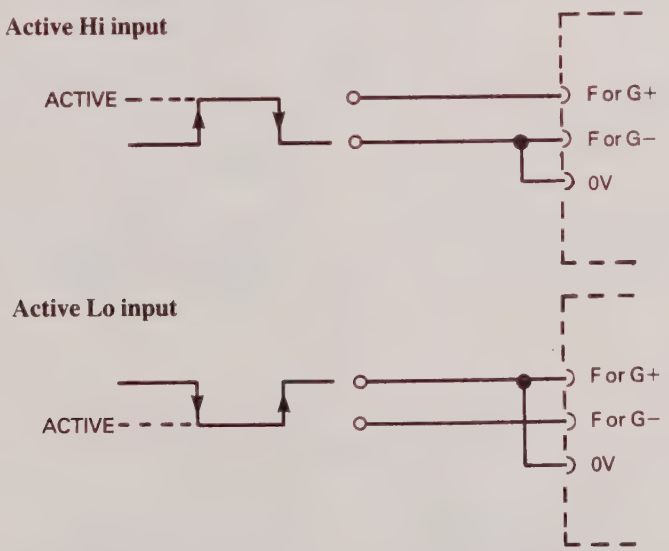
5 INPUT CONNECTIONS AND SWITCH SETTINGS



Depending on the input to be connected, see the appropriate diagram Fig. 7.1a/b, or Fig. 7.2a/b (pages 7.12 to 7.15), as indicated in the chart above. Each diagram shows how to connect a signal source which is floating, or earthed. The internal input circuitry within the Counter/Timer is also shown for general reference.

Switches 1 to 5, which correspond with Channels 1 to 5 respectively, determine bias conditions in the internal input circuit. Each switch must be correctly set for the type of input connected to each channel, and the appropriate setting is indicated in each of the diagrams Fig. 7.1a/b and 7.2a/b. Note that 2 wafers of each switch set up the G input, and the other 2 set up the F input. In each diagram both parts of the switch are shown set for the corresponding type of input, but in practice they may be set independently, i.e. G input may be set for TTL Active Hi, and the F input to TTL Active Lo. The physical location of switches S1 to S5 is shown in Fig. 7.3, page 7.16.

Simplified examples of Active Hi and Active Lo input connections are shown below.



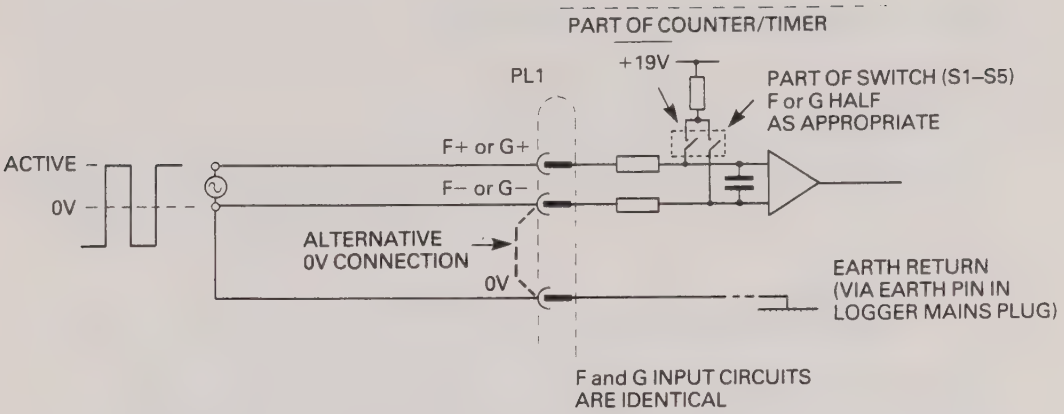
Arrows on the waveforms indicate the transitions detected in Interval measurement.

All input connections must be made to the appropriate pin in the sub-miniature D-type connector which mates with PL1 on the Counter Timer. Pin connection details for the connector/PL1 are given in Fig. 7.4 page 7.16.

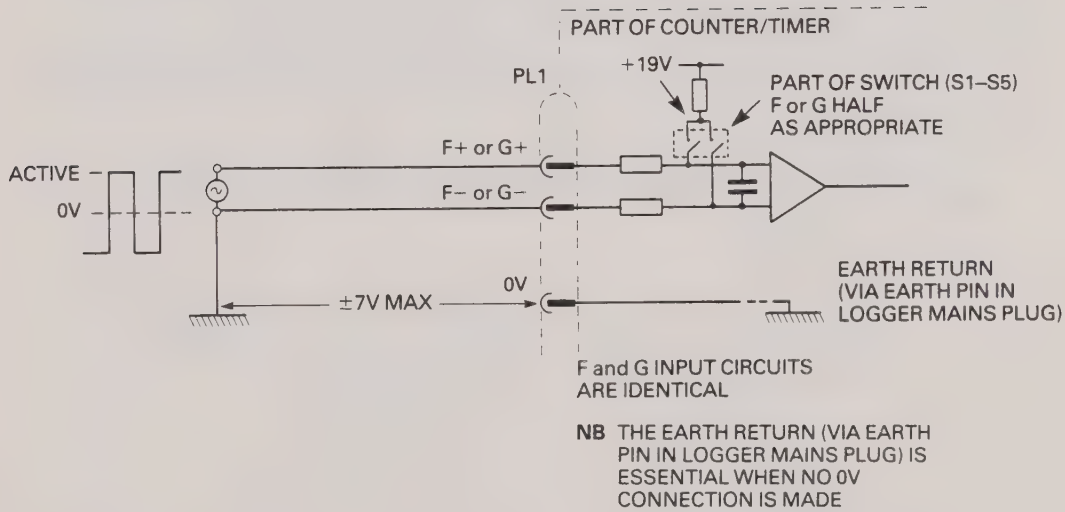
For reference purposes the input circuitry for a complete input channel is shown in Fig. 7.5, page 7.17. Channel 1 is shown, but the circuitry for all channels is identical.

DIFFERENTIAL INPUTS: Active Hi

Input connections for a floating signal source



Input connections for an earthed signal source



NB THE EARTH RETURN (VIA EARTH PIN IN LOGGER MAINS PLUG) IS ESSENTIAL WHEN NO 0V CONNECTION IS MADE

S1 to S5 Settings (for floating or earthed signal source)

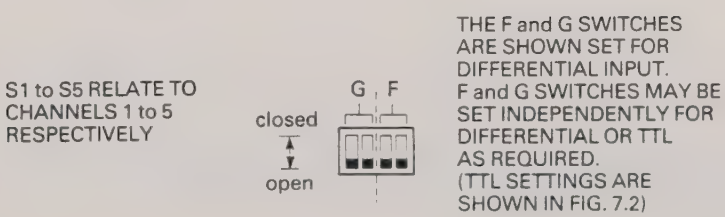
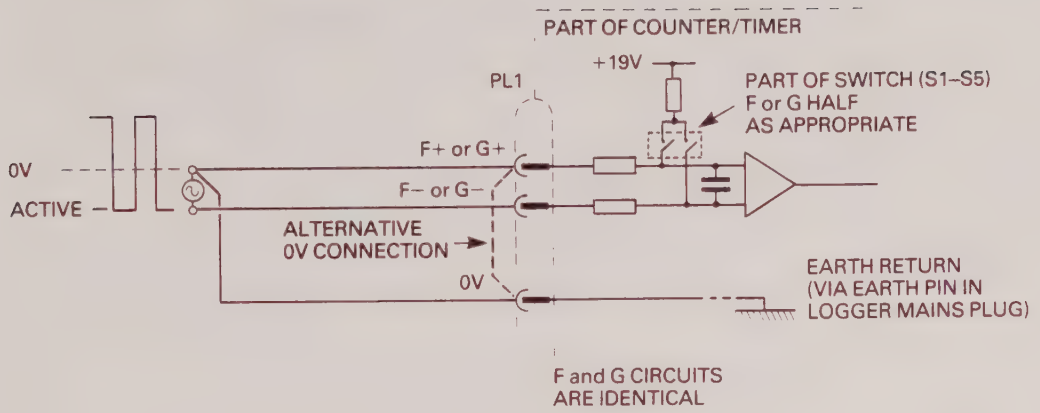


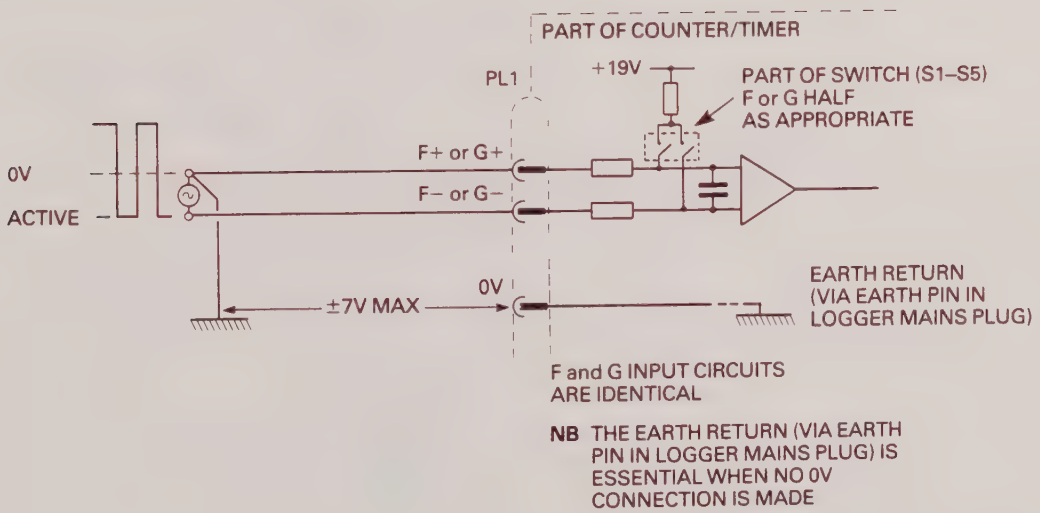
Fig. 7.1a *Differential Inputs: Active Hi*

DIFFERENTIAL INPUTS: Active Lo

Input connections for a floating signal source



Input connections for an earthed signal source



S1 to S5 Settings (for floating or earthed signal source)

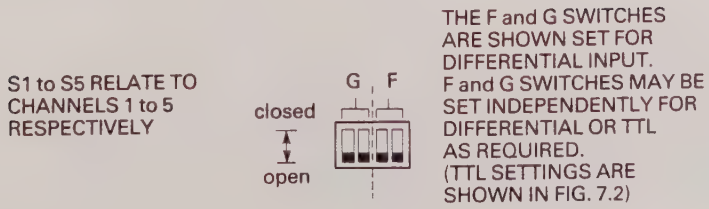
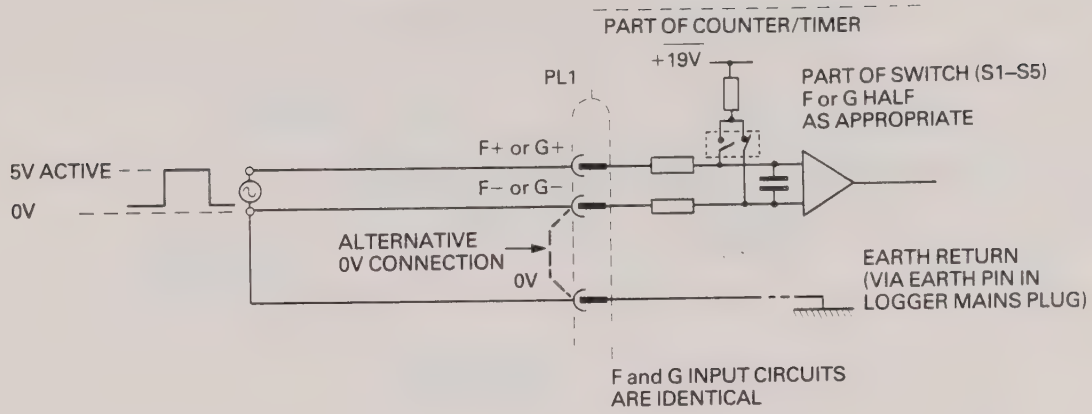


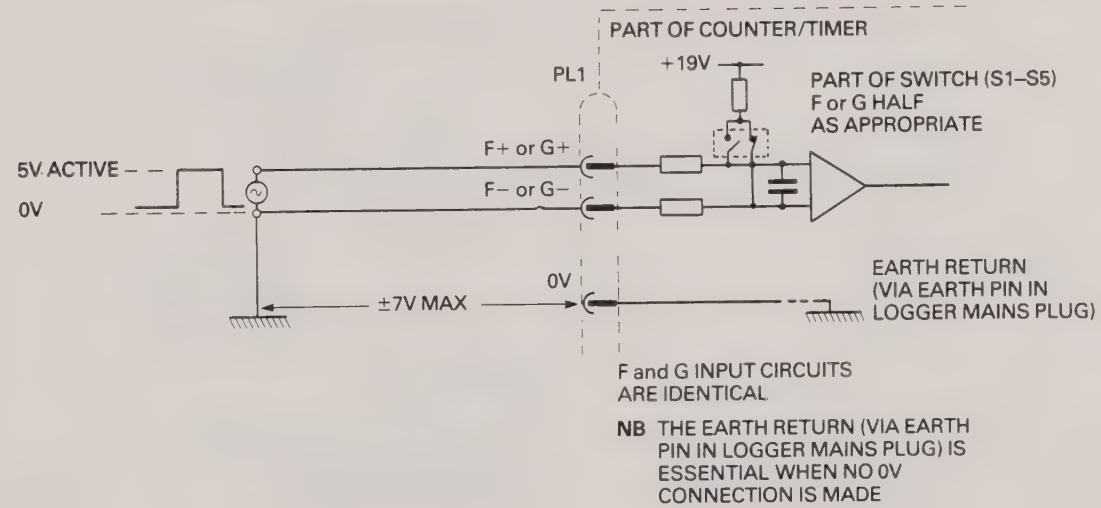
Fig. 7.1b *Differential Inputs: Active Lo*

TTL INPUTS: Active Hi

Input connections for a floating signal source



Input connections for an earthed signal source



S1 to S5 Settings (for floating or earthed signal source)

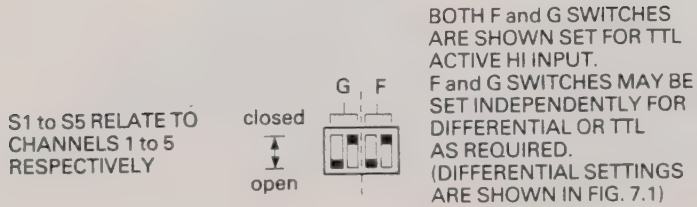
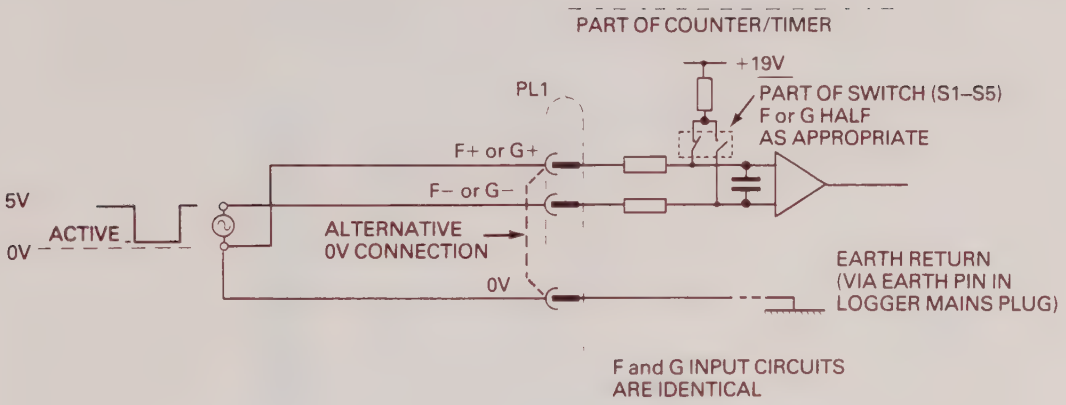


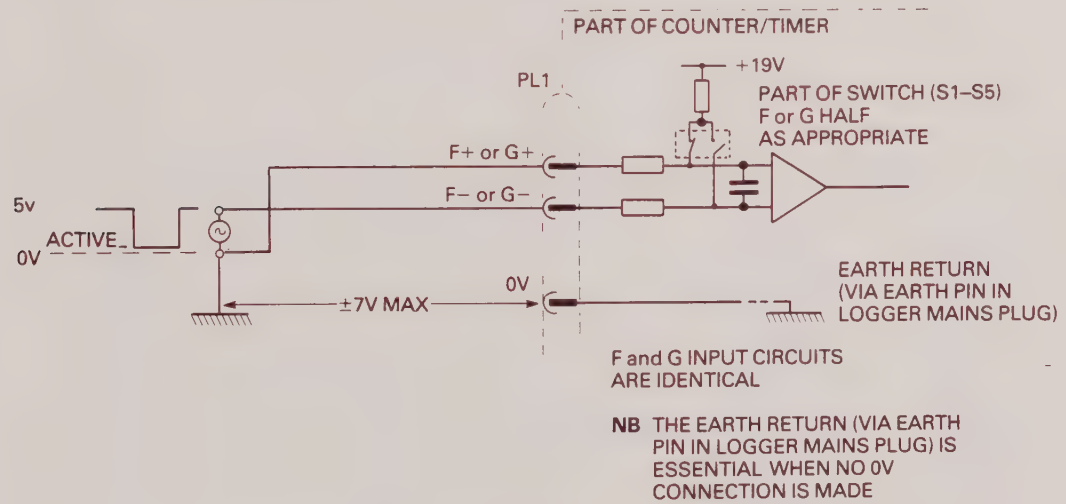
Fig. 7.2a *TTL Inputs: Active Hi*

TTL INPUTS: Active Lo

Input connections for a floating signal source



Input connections for an earthed signal source



S1 to S5 Settings (for floating or earthed signal source)

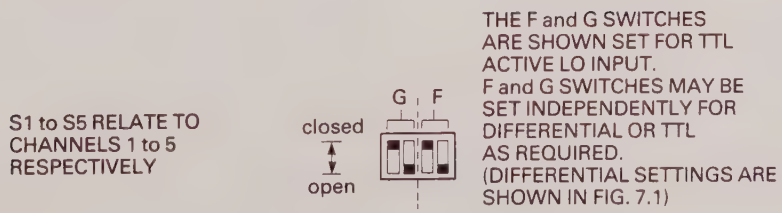


Fig. 7.2b *TTL Inputs: Active Lo*

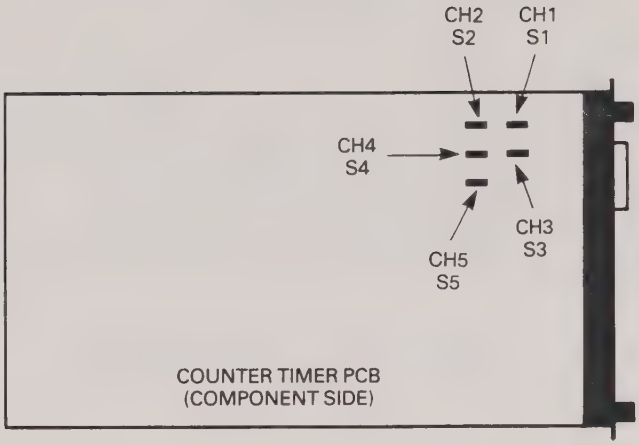
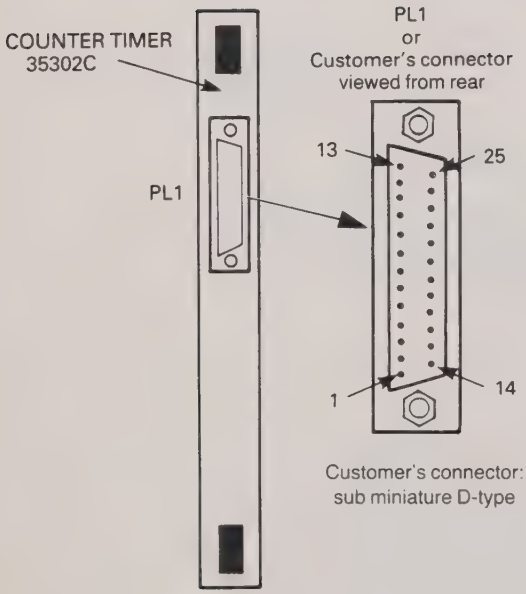


Fig. 7.3 *Location of switches S1 S5*



PL1 signals

PIN NO.	SIGNAL
1	F1−
2	G1−
3	F2−
4	G2−
5	F3−
6	G3−
7	F4−
8	G4−
9	F5−
10	G5−
11	+19V
12	0V
13	0V19
14	F1+
15	G1+
16	F2+
17	G2+
18	F3+
19	G3+
20	F4+
21	G4+
22	F5+
23	G5+
24	−30V } FOR TEST PURPOSES ONLY
25	+12V }

Fig. 7.4 *PL1 Pin Numbering*

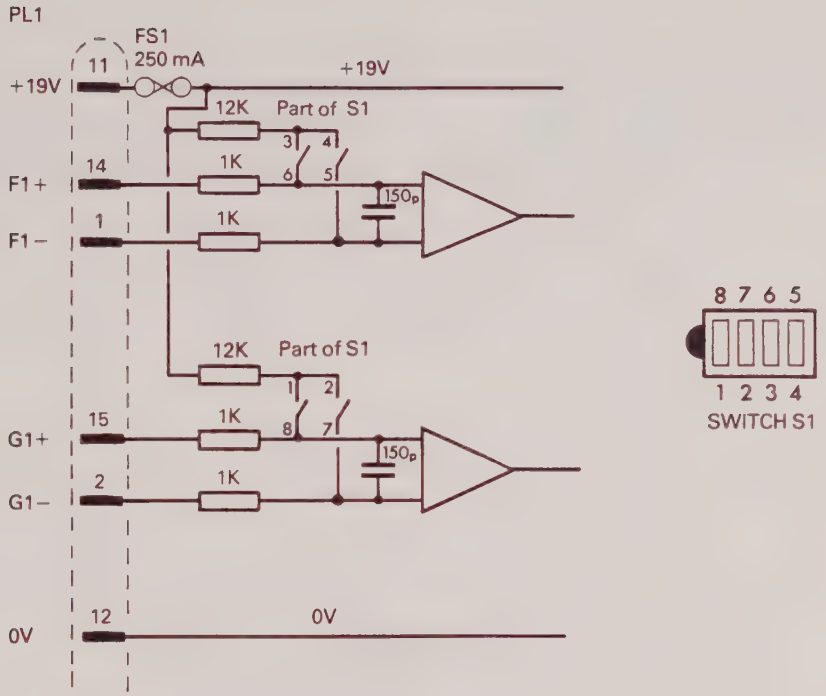


Fig. 7.5 *Input circuitry for Channel 1*

The input circuit for Channel 1 is shown above. All other channels (2 to 5) are identical; only the PL1 pin numbers and switch numbers will vary, i.e. for Channel 2, the inputs will be:

F2−
F2+
G2−
G2+
and the switch will be S2.

+19V (pin 11) and 0V (pin 12) will be common for all 5 input channels.

6 TRANSDUCER ENERGISATION

The 19V supply rail from the logger is available at PL1 pin 11, for energising transducers if required. The maximum current which may be drawn from this supply via any one Counter Timer Module is 250 mA (the supply is fused within the Module by a 250 mA fuse, see Fig. 7.5). However, if more than one Counter Timer is fitted to the logger, the maximum total current which may be drawn from this supply is:

400 mA, if the printer is to be used, or
1A, if the printer is never to be used.

7 GENERAL PROCEDURE FOR MAKING CONNECTIONS AND SWITCH SELECTION

1. Refer to the preceding sections for general information, as necessary:

		Page No.
1	Introduction	7.1
2	The Measurement Functions	7.2
2.1	Frequency	7.2
2.2	Period	7.5
2.3	Totalise	7.7
2.4	Interval	7.8
3	Obtaining Results	7.9
4	Using the F and G Inputs	7.10
5	Input Connections and Switch Settings	7.11
6	Transducer Energisation	7.17

2. Using the 25-way sub-miniature D-type connector supplied (which mates with PL1), connect the inputs as indicated:

   2.1 For Differential inputs, Active Hi, see Fig. 7.1a (page 7.12).
   For Differential inputs, Active Lo, see Fig. 7.1b (page 7.13).
   For TTL inputs, Active Hi, see Fig. 7.2a (page 7.14).
   For TTL inputs, Active Lo, see Fig. 7.2b (page 7.15).

   2.2 See Fig 7.4 (page 7.16) for sub-miniature D-type connector pin numbering.

   2.3 Ensure that the F and G inputs for each channel are correctly used for the measurement function required, see 'Using the F and G inputs', page 7.10.

   2.4 When measuring high frequency signals, keep connecting leads as short as possible. If long leads are used, the effects of cable capacitance and source impedance may cause degradation of the signal. This may cause the signal to be attenuated, and fast edges may be slowed down thereby affecting timing of the pulses.

3. With the logger switched off, remove the Counter Timer Module from the mainframe.

4. Set switches S1 to S5 to suit the signals being connected; the location of the switches is shown in Fig. 7.3 (page 7.16) and the switch settings are shown in Figs. 7.1a or b, or 7.2a or b.

5. Refit the Module in the mainframe and fit the sub-miniature D-type connector to PL1.

6. With the logger switched on, select Channel Definition and set-up the Counter Timer channels as required. Note that the channel numbers for a Counter Timer Module will be:

   1 to 5, if fitted in Slot 4
   21 to 25, if fitted in Slot 5
   ⋮
   181 to 185, if fitted in Slot 13.

7. Incorporate the Counter Timer channels into the Task(s) as required, using Task Definition.

8. Two simple examples, showing input wiring, switch settings and signal timing diagrams are shown on pages 7.19 and 7.20.

9. Switch off Counter Timer channels, when they are not being used. This is done using Channel Definition ('SENSOR, Select Measurement Type, OFF'). Unnecessary programming overheads are minimised when the unwanted channels are switched off, and maximum efficiency is thereby achieved within the logger.

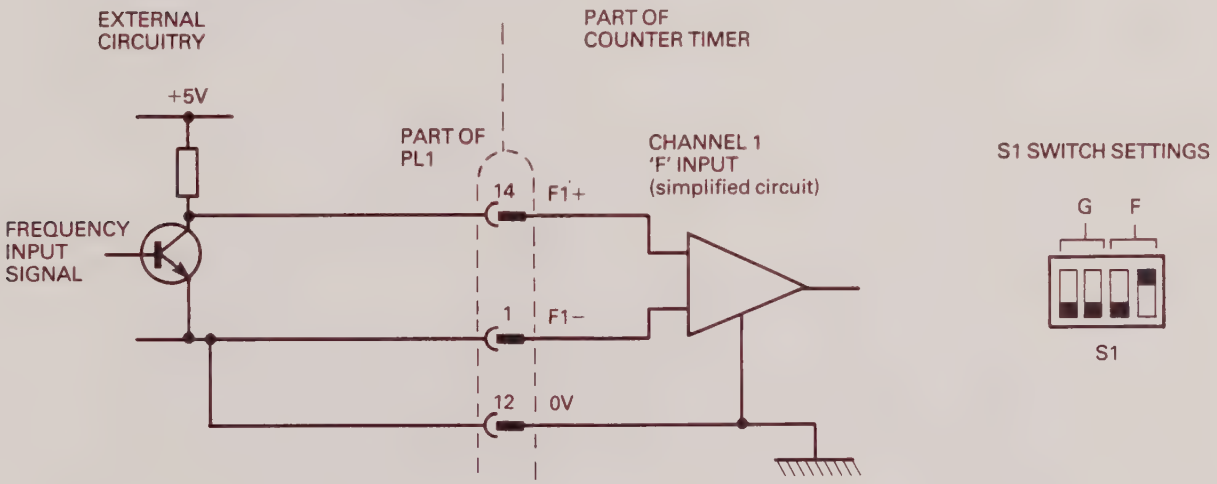
Error Messages: 'E8 Wait' is output if no result has yet been accumulated (e.g. in Frequency, if gate time had not finished when scan occurred).

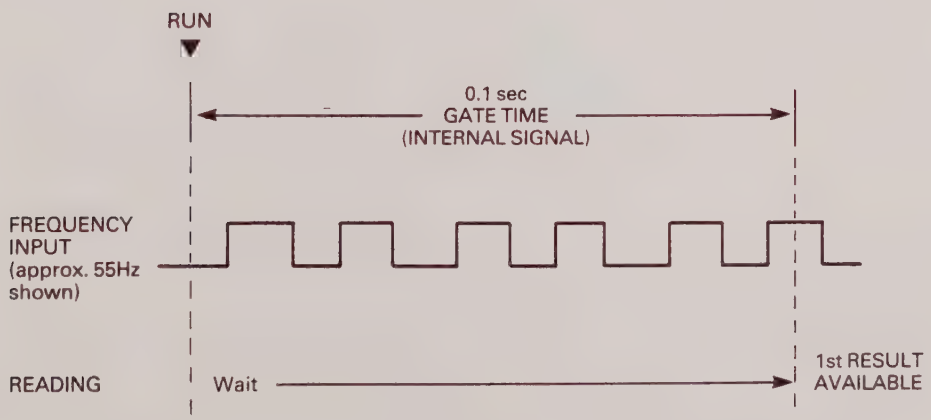
'E9 Off' is output if Counter Timer channel is switched off.

Example 1

Frequency measurement, on Channel 1, using internal gating TTL input, Active Hi.

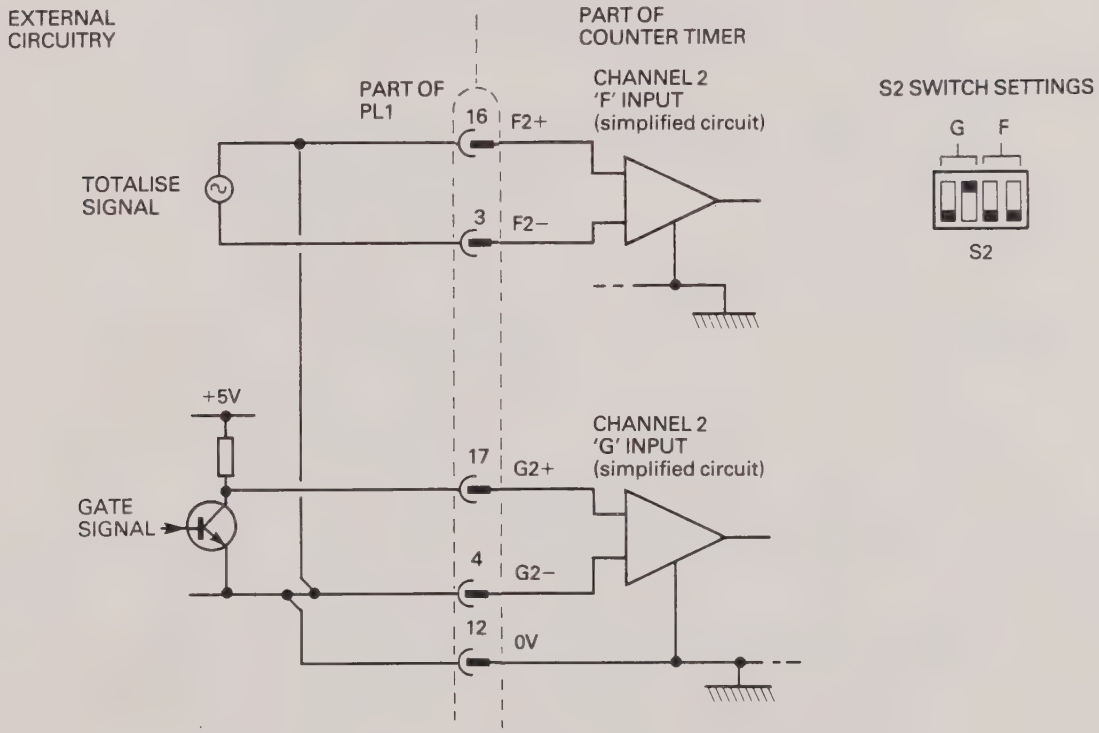
Internal gate is set to 0.1s.

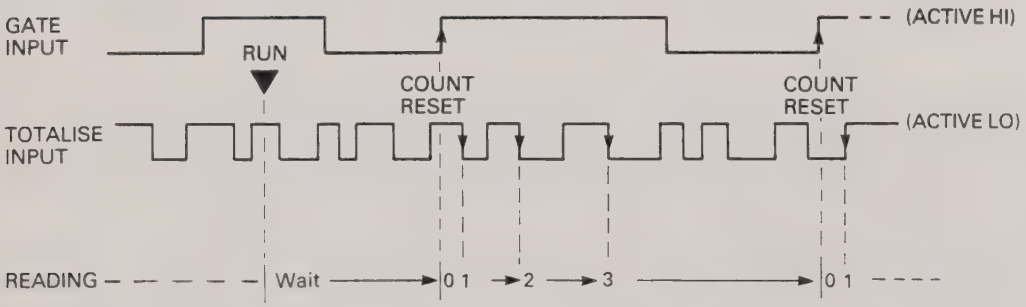




Example 2

Totalise measurement on Channel 2.
Differential input for signal (F+ and F−), Active Lo
TTL input for gate (G+ and G−), Active Hi.





Chapter 8
Using the Analogue Output Module 35301E

Contents

	Page No.
1. Introduction	8.1
2. Setting up and Using the Outputs	8.2
3. Current Outputs: Connections and Switch Settings	8.4
4. Voltage Outputs: Connections and Switch Settings	8.5
5. Current/Voltage Selector Switches	8.6
6. PL1/PL2 Connections	8.7
7. Procedure	8.8

1 INTRODUCTION

The Analogue Output Module has five output channels which are intended for driving external devices such as plotters and chart recorders. Each of the channels is primarily intended to produce a scaled voltage or current output which is proportional to the measured result from a nominated input channel; this is activated using the SCALE facility within the Channel Definition. To help set up or test an external device, each output can be set manually to a specified value; this is achieved using the VALUE facility within the Channel Definition.

Each of the five output channels is a fully floating voltage or current source. The range of voltage or current which may be output is indicated below.

	Min = 0%	Max = 100%
Voltage	0V	10V
Current	4mA	20mA

Each channel can be independently set for voltage or current output, and selection is achieved by means of switches on the pcb.

The preferred location for an Analogue Output Module in the ORION frame is indicated on page 1.17. Note that the combined total of Analogue Output and Counter Timer Modules must not exceed five per frame.

NB The Analogue Output Module can only be used with an ORION which has a modification status of B3 or higher. The modification status is indicated on the Serial Number Label which is located under the right-hand carrying handle (as viewed from front of instrument). If the status is lower than B3, consult a Solartron Service Department.

2 SETTING-UP AND USING THE OUTPUTS

Firstly, each output must be set-up as required, using Channel Definition for the output channel. For normal use select SCALE, or for setting-up or test purposes only, select VALUE.

SCALE — Enter two values which represent measurements obtained from the nominated input channel.
1. The input measurement which will give a minimum (= 0%) output
2. The input measurement which will give a maximum (= 100%) output

This sets up the scaling and offset for the output channel.

VALUE — Enter the set value (percentage) of output required.
(This facility is intended solely for setting-up and testing purposes).

If SCALE was selected, ensure that the number of this output channel is entered in the Channel Definition for the associated input channel (under 'OUTPUTS, enter analogue output channel'). This links the nominated input with the output channel. Thereafter, whenever that input channel is scanned or monitored, the result obtained will cause the output channel to produce a proportional voltage or current output, within the scaling limits (0% and 100%) previously entered. This output level will be maintained until it is updated by another measurement, from a scan or by monitoring, see diagram below. The output may also be changed manually using the VALUE facility at any time.

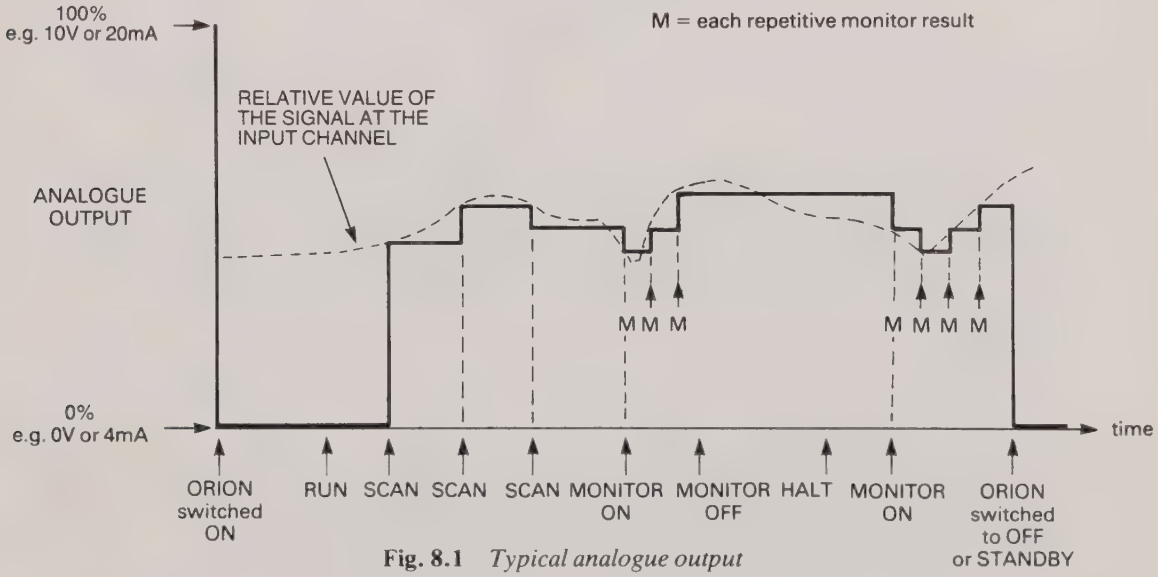


Fig. 8.1 *Typical analogue output*

Note that the 0% output level (0V or 4mA) is maintained from switch on, until the first result becomes available. The output level obtained from a scan or monitor result will be maintained until updated by the next scan or monitor measurement. The output is unaffected by the Halt command or by monitor being switched off. Since monitoring produces repetitive measurements, each measurement (shown as 'M' in the diagram) will produce an updated output.

The output channel can be updated at a maximum rate of approximately 3 times/second. The output cannot be updated faster than approximately once every 300ms.

Note that the scan or monitor measurement of the input channel, which is used to control the output, is that given after engineering units (e.g. °C, μstrain), or a Conversion has been applied to the measurement.

If the measurement is to be processed before being used to control the output, the data must first be copied from the Channel variable to another variable, as described in Chapter 12. The data in the new variable can then be processed as part of an Output Statement.

Note that an Analogue Output can either be controlled via the Channel Definition Output soft key or by a Processing Output Statement. The two methods must not be used on the same Analogue Output simultaneously.

The scaling (SCALE) facility, in which the maximum (100%) and minimum (0%) measurements are defined, permits maximum resolution of results within a particular range of interest. This is illustrated in the example below.

An Analogue Output channel is connected to a plotter, and is being used to produce a plot of temperature, as measured by the associated input channel. (This input has a thermocouple connected). The range of temperature which is likely to be measured is between 50°C and 85°C. Therefore the SCALE '0%' value is set to 50 (= 50°C) and the '100%' value is set to 85 (= 85°C). This effectively applies a scaling and offset to the output, as shown in Fig. 8.2.

Note that Fig. 8.2 shows the relationship between the output voltage or current and the input channel measurement; the output channel itself will always produce a fixed output level, being updated by each input measurement, as shown in Fig. 8.1.

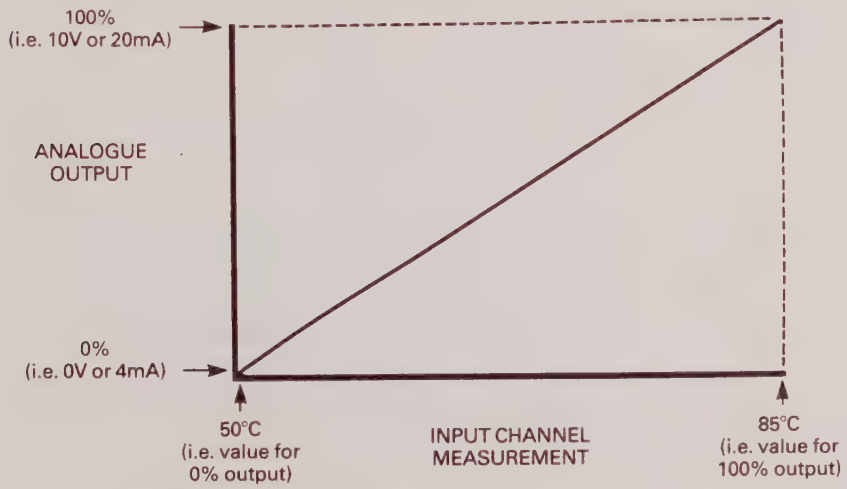


Fig. 8.2 *Relationship between analogue output and measured input for the example given*

In this example, the entire range of the output is thereby used to indicate measured temperatures between 50°C and 85°C, thus giving maximum output resolution over this selected input range.

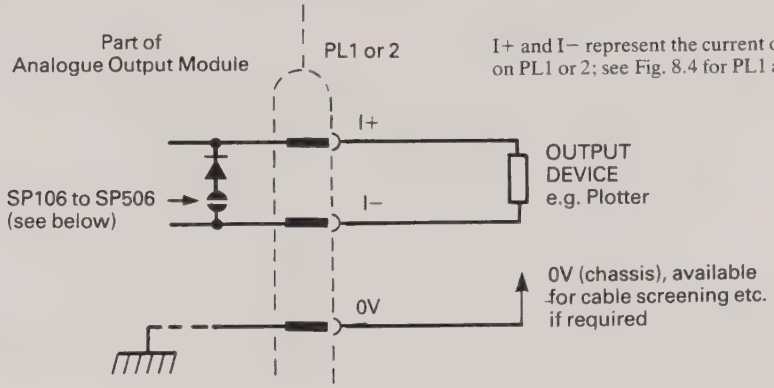
The SCALE 0% and 100% values can be set as required for any given application. For example, to produce a direct scaling factor with no offset, set the 0% value to zero input (e.g. 0 μstrain, from a strain gauge input channel) and the 100% value to the maximum level expected (e.g. 250 μstrain). To produce a negative scaling factor make the 100% level lower than the 0% value.

The actual value of the output may be inspected at any time by monitoring the output channel, using the MON facility. Alternatively, the value may be obtained by selecting the Channel Definition for the output, and pressing 'VALUE'. The present output level will be indicated, irrespective of whether it was set using VALUE or by a measurement of the associated input channel using the SCALE facility.

It is worth noting that analogue outputs may also be controlled via one of the ORION's interfaces, by sending a command to ORION in the normal way. Thus, a computer which may be processing results obtained from measurements can cause an analogue output channel to change state.

3 CURRENT OUTPUTS: CONNECTIONS AND SWITCH SETTINGS

Connections



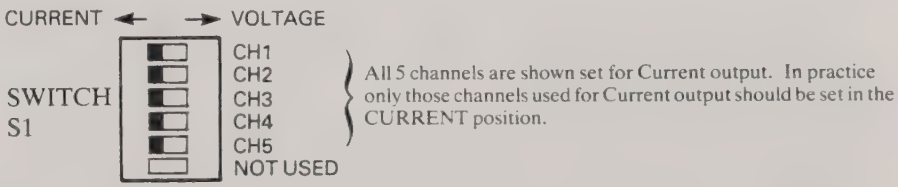
Output Current Range: 4mA to 20mA
(4mA = 0%, 20mA = 100%)

Output Voltage Compliance: 10V
i.e. voltage across the output device must not be allowed to exceed 10V, for correct current output.

SP106 to SP506: Split pads SP106 to 506 relate to channels 1 to 5 respectively. The Analogue Output Module will be supplied with all links connected. Each channel thus has the catching diode connected across the current outputs, giving protection against back-emf's when using inductive loads. Any link may be disconnected if required. Links SP106 to 506 are located on the pcb close to S101 to 501 respectively.

Switch Settings

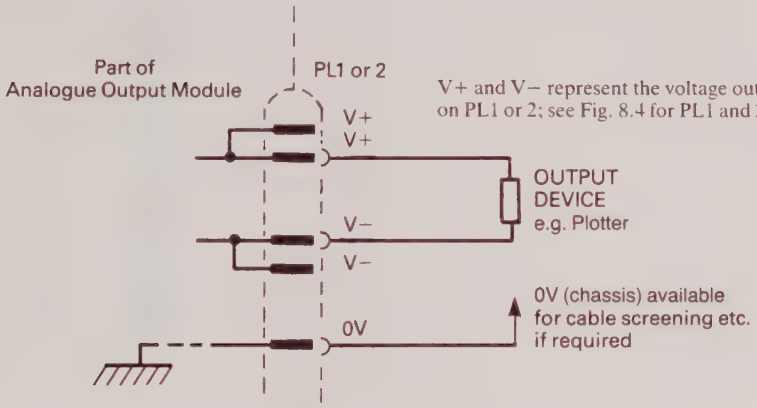
*S101 to S501 relate to Channels 1 to 5 respectively (each channel may be independently set for Current or Voltage output).



NB Ensure that the Current/Voltage settings of S101 to S501 do not conflict with the corresponding Current/Voltage setting on S1 for each respective channel CH1 to CH5.

4 VOLTAGE OUTPUTS: CONNECTIONS AND SWITCH SETTINGS

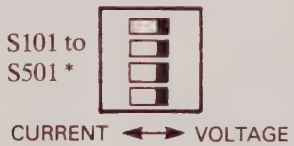
Connections



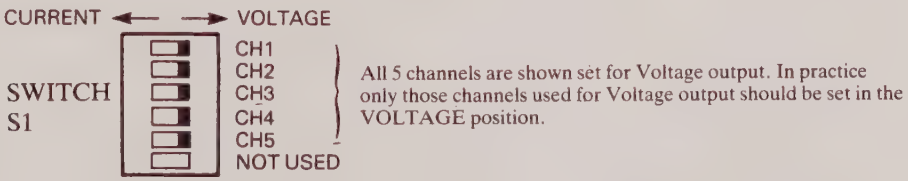
Output voltage range: 0V to 10V
(0V = 0%, 10V = 100%)

Output Current: 0mA to 20mA

Switch Settings



*S101 to S501 relate to Channels 1 to 5 respectively (each channel may be independently set for Current or Voltage output).



NB Ensure that the Current/Voltage settings of S101 to S501 do not conflict with the corresponding Current/Voltage setting on S1 for each respective channel CH1 to CH5.

5 CURRENT/VOLTAGE SELECTOR SWITCHES

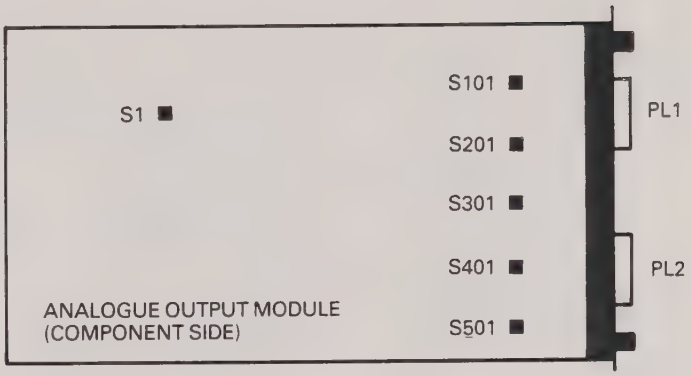
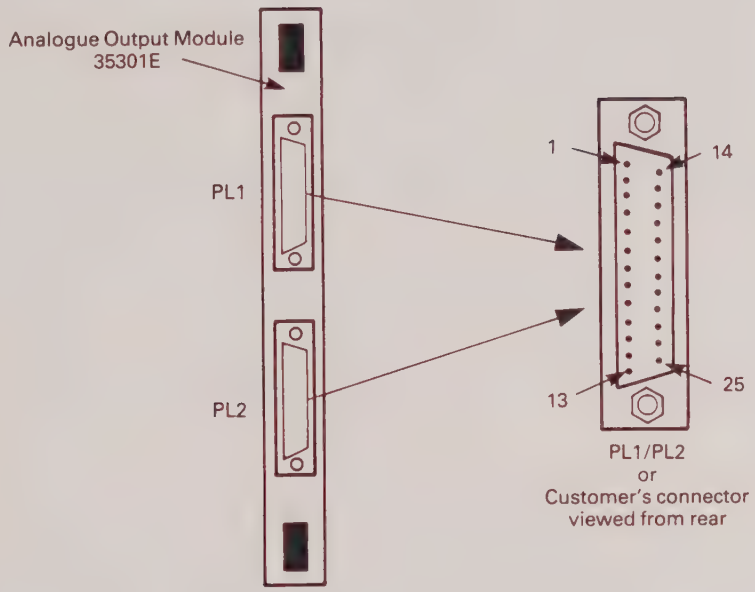


Fig. 8.3 *Analogue Output Module: Location of Switches*

VOLTAGE/CURRENT SELECTION	
Switch No.	Channel
S101	1
S201	2
S301	3
S401	4
S501	5
S1	1 to 5

NB Ensure settings of S101 to 501 do not conflict with those of S1.

6 PL1/PL2 CONNECTIONS



PL1: Contains Channels 1-3

PL1 PIN No.	SIGNAL CH1	CH2	CH3
1	I+		
2	V+		
3	V+		
4			
5		I+	
6		V+	
7		V+	
8			
9			
10			I+
11			V+
12			V+
13			
14	I−		
15	V−		
16	V−		
17			
18		I−	
19		V−	
20		V−	
21			
22			
23			I−
24			V−
25			V−

PL2: Contains Channels 4 and 5

PL2 PIN No.	SIGNAL CH4	CH5	0V
1	I+		
2	V+		
3	V+		
4			
5			
6			
7			0V
8			
9			
10		I+	
11		V+	
12		V+	
13			
14	I−		
15	V−		
16	V−		
17			
18			
19			0V
20			0V
21			
22			
23		I−	
24		V−	
25		V−	

Fig. 8.4 *PL1 and PL2 Pin Numbering*

7 PROCEDURE

1. Refer to the preceding sections for general information, as necessary:

	Page No.
1. Introduction	8.1
2. Setting-up and Using the Outputs	8.2
3. Current Outputs: Connections and Switch Settings	8.4
4. Voltage Outputs: Connections and Switch Settings	8.5
5. Current/Voltage Selector Switches	8.6
6. PL1/PL2 Connections	8.7

2. Using the 25-way sub-miniature D-type connectors supplied (these mate with PL1 and PL2), connect the output devices, as follows:

   2.1 For Current output channels, connect as shown on page 8.4.
   2.2 For Voltage output channels, connect as shown on page 8.5.
   2.3 See page 8.7 for PL1/PL2 pin connections.

3. With the logger switched off, remove the Analogue Output Module from the mainframe.

4. Set switches S101 to 501, and S1 as appropriate:

   4.1 See page 8.6 for location of switches.
   4.2 For output channels 1 to 5, set S101 to S501 respectively, to either CURRENT or VOLTAGE; these settings are shown adjacent to the connection diagram for current (page 8.4) or voltage (page 8.5).
   4.3 Set S1 (CH1 to CH5) to correspond with the CURRENT/VOLTAGE settings of S101 to S501; S1 settings are shown adjacent to the connection diagram for current (page 8.4) or voltage (page 8.5).

5. For current outputs only, see reference to Links SP106 to SP506 on page 8.4, and connect/disconnect links as required.

6. Refit the Module in the mainframe, and fit the sub-miniature D-type connectors to PL1 and PL2.

7. With the logger switched on, select Channel Definition and set up the Analogue Output channels as required. Note that the output channel numbers will be:

   1-5, if fitted in Slot 4
   21-25, if fitted in Slot 5
   ⋮
   181-185, if fitted in Slot 13

8. Ensure that the Channel Definition for each associated input channel contains the appropriate output channel number, i.e. under 'OUTPUT, enter analogue output channel'.

   Note: the first time that OUTPUT is pressed, the Display will request the O/P Channel numbers for the digital **Alarm** outputs. By pressing ENTER, the Display changes to request:

   "Enter analogue output channel"

9. Incorporate the associated input channel(s) into one or more Tasks as required, using Task Definition. Note that neither Event inputs nor Bit Status inputs can be applied to an Analogue Output channel.

Chapter 9
Front Panel Controls

Contents

		Page No.
1	Introduction	9.2
2	Location of Controls	9.2
3	Function Keys	9.3
3.1	Control Keys	9.4
	RUN	9.4
	PAUSE	9.4
	MONitor	9.4
	LOCAL	9.5
3.2	Definition Keys	9.5
	CHANnel	9.5
	TASK	9.11
	CONVersion	9.14
	TEXT	9.14
	INITialise	9.15
	TAPE	9.15
	LIST	9.17
	PAPER ADVance	9.17
3.3	Edit Keys	9.17
	↑ ↓	9.17
	← →	9.18
	CHARacter INSERT	9.18
	CHARacter DELETE	9.18
	LINE INSERT	9.18
	LINE DELETE	9.18
3.4	Processing Keys — see Chapter 12	
4	Keypad	9.19
5	Soft Keys	9.20
6	Keyswitch	9.21
7	Display	9.21
8	Scrolling Facility	9.22
9	Power Annunciators	9.22
10	Audible Indications	9.23

1 INTRODUCTION

This chapter indicates how the front panel keys may be used to control and set-up an ORION *delta*. Each Function Key permits a related group of operating parameters to be set-up, changed or examined. When controlling or setting-up an ORION *delta* or ORION F from a remote device (e.g. a vdu), the information is input in similar related groups, using the 'command language' defined in the accompanying handbook 'Communicating with ORION *delta*'. Thus, the general information given under each Function Key heading in this chapter is equally applicable for remote operation of an ORION *delta* or F.

2 LOCATION OF CONTROLS

The front panel controls are arranged in 4 groups:

FUNCTION KEYS
KEYPAD
SOFT KEYS
KEYSWITCH

The front panel, which also includes the 2-line DISPLAY and the POWER ANNUNCIATORS, is illustrated in Fig. 9.1.

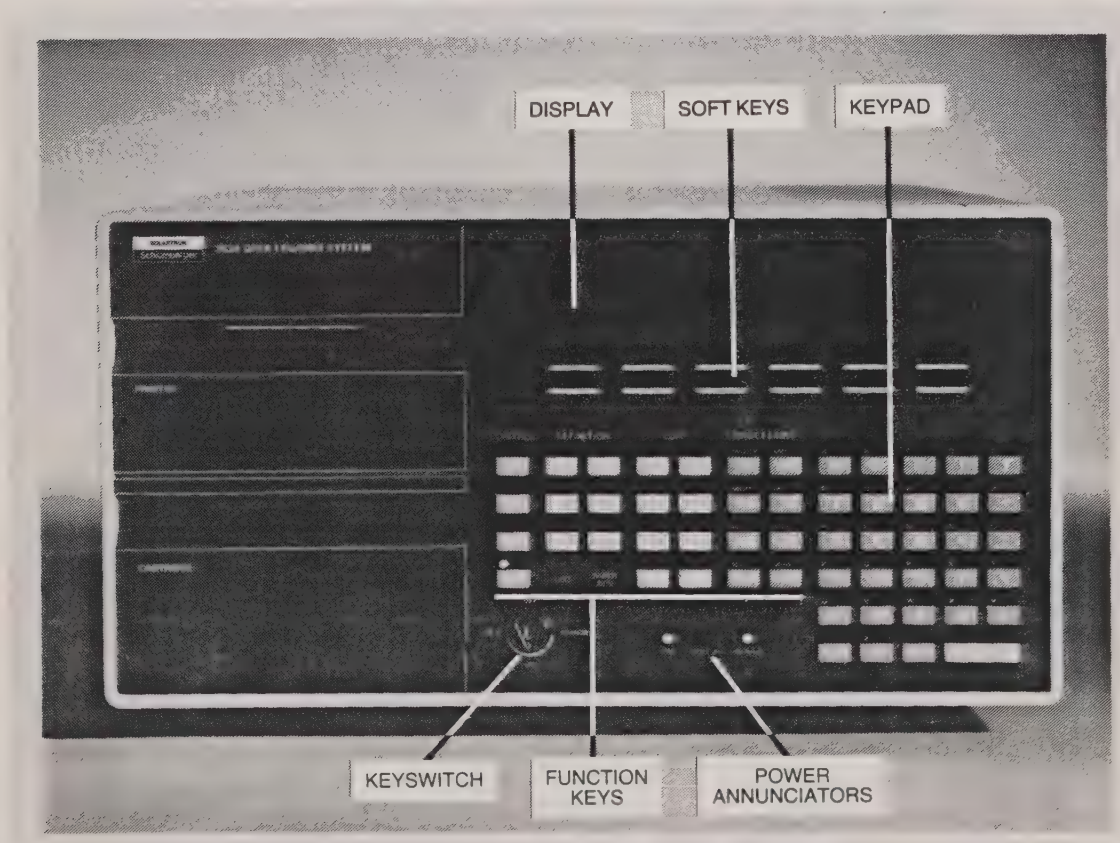


Fig.9.1 *The Front Panel*

Full control and programming of the logger is achieved at the front panel. The Function Keys and the Keypad are used in conjunction with the Soft Keys which have variable meaning depending on the function selected. The meaning of each soft key at any time is indicated on the lower line of the display. The use of soft keys simplifies operation of the logger and necessitates only the minimum of Function Keys.

This Chapter acts as a reminder for the user who already has some familiarity with ORION. A full explanation of certain functions will be found in other Chapters where indicated.

3 FUNCTION KEYS

The FUNCTION keys are sub-divided into 4 categories:

3.1 CONTROL:

RUN
PAUSE
MONitor
LOCAL

3.2 DEFINITION:

CHANnel
TASK
CONVersion
TEXT
INITialise
TAPE
LIST
PAPER ADVance

3.3 EDIT:

↑
↓
←
→
CHARacter INSERT
CHARacter DELETE
LINE INSERT
LINE DELETE

3.4 PROCESSING:

IF/TRIGGER
FOR/AND
THEN/ABORT
TO/OR
ELSE/KILL
NEXT/STEP
ENDIF/SET
PRINT/GOTO

Only used for processing programs, see Chapter 12

3.1 CONTROL KEYS

Permits the programmed tasks to be stopped and started.

RUN (annunciator illuminated)
All programmed tasks are operational, logger awaits the first valid trigger. Programs can be examined but not changed.

HALT (annunciator not illuminated)
All tasks stop; setting-up information can be entered or changed.

Intended to stop logging temporarily to printer or cartridge.

PAUSE (annunciator illuminated)
All programmed tasks are operational except that no data is output to any device or interface. This facility is primarily intended for use when changing the paper roll. When the task trigger source is set to 'continuous', PAUSE will not only stop the output but will also stop the scan.

CONTINUE (annunciator not illuminated)
Cancels the PAUSE state, all programmed tasks are fully operational.

Select monitor data:	Chnls	Enter Channel Numbers to be monitored, a Monitor scan is initiated and the results are output. See 'Entering Channel Numbers', pages 9.5 and 9.12. Do not Monitor channels used for event counting (Counter).
	Task	Enter Task Number to be monitored. Results are output when task is running.
	Headrs	Scan Task n (STn) and time of start, and Done Task n (DTn), are output at beginning and end of scan respectively.
	On	Select On, when all Monitor parameters have been set-up, to start monitoring of Channels, Task or Headers.
	Off	Monitoring stops but all parameters remain in set-up state.

Select output device:	Displ	To built-in display	
	RS232	To device connected to PL1 on Controller.	
	Print	To built-in printer	
Select format required:	Full	(header includes title)	overrides Format selection in Task Definition
	Compact	(no title in header)	

Press to request local (front panel) operation when logger is being controlled remotely. Annunciator On = LOCAL.

At switch-on, logger assumes LOCAL state.

3.2 DEFINITION KEYS

Channels Enter Channel Numbers.

The following rules must be observed:

a. Up to 18 individual Channels (e.g. 1, 2, 20, 35, etc), or:

b. Up to 6 groups of Channels (e.g. 1-20, 21-40, etc), or:

c. A mixture of the above, may be entered.

3 single Channels occupy the same space as 1 group.

When more than 40 characters are selected the scrolling facility automatically operates, see Section 8.

Channel Numbering when using Slave Expansion Units

When a Main Unit is used with Slave Expansion Units the following Channel numbering applies:

	Channel Numbers
Main Unit	001 to 200
1st Slave Unit	201 to 400
2nd Slave Unit	401 to 600

Any of the Channels 001 to 600 may be incorporated in any one Task.

When defining any one Channel group in a Channel Definition, note that Channels can only be grouped if all Channels in the group are physically located within one Unit, i.e. all in the Main Unit, or all in 1st or 2nd Slave. It is not permissible for the Channels in any one group to be contained in 2 or more Units.

For example, it is not permissible to group Channels 185-235, since some are contained in the Main Unit and some in the 1st Slave. These Channels should be entered in the Channel Definition in two groups thus:

185-200, 201-235

Similarly, Channels 395-450 cannot be entered in one group, since some are contained in the 1st Slave and some in the 2nd Slave. These Channels should be entered thus:

395-400, 401-450

Channels can be grouped, in which case all Channels should be of the same general type (e.g. all strain gauges, all thermocouples, or all digital outputs, etc.)

All Channels must be **one** of the following types:
analogue inputs,
or event/status inputs,
or digital inputs
or counter/timer inputs

A mixture of types is not permitted in any one Channel Definition.
Channel Definition does not apply to digital outputs.

When the Channel Numbers have been entered the logger automatically offers the appropriate Channel Definition setting-up procedure, depending on the type of Channels selected.

The setting-up details for each type of input/output Channel are given on the following pages, as indicated in the table below.

CHANNEL TYPE	MODULE	PAGE NUMBER
Analogue inputs	Reed Selector 35301A or Solid State Selector 35301B	9.6 below
Event/Status inputs	Event/Status Selector 35302A	9.8
Digital inputs/outputs	Digital I/O Selector 35302B	9.9
Counter Timer inputs	Counter Timer Module 35302C	9.10
Analogue Outputs	Analogue Output Module 35301E	9.10

Sensor
Select Measurement type:

Volts Amps (current)	DC or AC measurement
Ohms (resistance)	4-or 2-Terminal Measurement energising current: 1mA, 100μA or user-defined, thermal drift compensation: on/off
Thcpl	Thermocouple Type: E, J, K, R, S or T Cold Junction compensation: Ambient, Reference 1–4, External 0°C (Use INITIALISE 'References' to set temperature of Ref 1–4).
PRT	Platinum Resistance Thermometer Select thermal drift compensation: off/on
Strain	Strain Gauge.

If Strain is selected:

Select bridge configuration

Full Br	Select thermal drift compensation: off/on, enter gauge factor, see page 4.4.
½ Br	Select thermal drift compensation: off/on, 8mA, 1mA or user-defined energisation current; enter gauge factor, see page 4.4.
¼ Br	Select thermal drift compensation: off/on, dummy: remote 120Ω, or user-defined dummy X, Y or Z. 8mA, 1mA or user defined energisation current, enter gauge factor, see page 4.4.

Conv

Enter conversion number (1 to 40), to apply conversion to all logged results

Alarm

Select alarm type:

Limits	Enter high and low limits
↑ Thres	Enter threshold value and lower limit of deadband.
↓ Thres	Enter threshold value and upper limit of deadband.
Change	Enter change size
Off	Alarm off, but any parameters set-up remain in memory.

Output

Enter 2 digital output channel numbers; (lo, hi).
On low alarm 'lo' goes from logic 0 to 1.
On high alarm 'hi' goes from logic 0 to 1.

Enter analogue output channel (Analogue Output Module).

Text

Enter text number (1-40), for text to be output with logged (non-alarm) data.

Enter alarm text number (1-40), for text to be output only with alarm data.

If Status/Event channels are entered

Select item to be defined:

Sensor

Select Digital Input Function:

bit	For individual or grouped status inputs: enter number of bits to be grouped (1-8).
BCD	For binary coded decimal (8-4-2-1) inputs: enter number of bcd decades (1-6).

inv BCD	For inverted logic binary coded decimal (8-4-2-1) inputs; enter number of bcd decades (1-6).
Event	Channel(s) continuously monitored for 'events'
Count	Input pulses counted; count reset between readings
Total	Input pulses counted; the count is cumulative (totalised) (max total = 65535)

Select transition of interest
Hi-Lo
or
Lo-Hi
or
Any

Conv

Enter conversion number (1 to 40), to apply conversion to all logged results.

Alarm (for all Sensor types except 'bit')

Select alarm:

Limits	Enter high and low limits
↑ Thres	Enter threshold value and lower limit of dead band.
↓ Thres	Enter threshold value and upper limit of dead band.
Change	Enter change size
Off	Alarm off, but any parameters set-up remain in memory.

Alarm (for 'bit' Sensor type only)

Select alarm on pattern match/mismatch (see Chapter 5, section 1.5):

Match	Alarm when input bit pattern is same as alarm pattern.
MisMch	Alarm when input bit pattern is different from alarm pattern.
Off	Alarm off, but any parameters set-up remain in memory.

Enter alarm pattern (1-8 bits). To mask any bit, enter [*] (=dont care); the corresponding Channel is output normally but disregarded by alarm detector.

Output

Enter 2 digital output Channel numbers: (lo, hi).
On low alarm 'lo' goes from logic 0 to 1.
On high alarm 'hi' goes from logic 0 to 1.
For Match/Mismatch alarms enter the number as n.n.
Enter analogue output Channel (Analogue Output Module).

Text

Enter text number (1-40), for text to be output with logged (non-alarm) data.
Enter alarm text number (1-40), for text to be output only with alarm data, or events.

If Digital Input/Output Channels are entered

Select item to be defined:

Sensor
Select digital input type:

bit	For individual or grouped status inputs, enter number of bits to be grouped (1-8).
BCD	For external-module-measurement mode* inputs, 8-4-2-1 binary coded decimal; enter number of bcd decades (1-6).
inv BCD	For external-module-measurement mode* inputs, $\bar{8}$-$\bar{4}$-$\bar{2}$-$\bar{1}$ inverted logic binary coded decimal; enter number of bcd decades (1-6).
binary	For external-module-measurement mode* inputs, enter number of 8 bit bytes. *The number of the module to be read (1-20), in external-module-measurement mode, is included in the Task Definition.

Conv
Enter conversion number (1-40), to apply conversion to all logged results.

Alarm (for all Sensor types except 'bit')
Select alarm type:

Limits	Enter high and low limits
↑ Thres	Enter threshold value and lower limit of dead band.
↓ Thres	Enter threshold value and upper limit of dead band.
Change	Enter change size
Off	Alarm off, but any parameters set up remain in memory.

Alarm (for 'bit' Sensor type only)
Select alarm on pattern match/mismatch:

Match	Alarm when input bit pattern is same as alarm pattern.
MisMch	Alarm when input bit pattern is different from alarm pattern.

Enter alarm pattern (1-8 bits). To mask any bit, enter [*] (=dont care); the corresponding Channel is output normally but disregarded by alarm detector.

Off	Alarm off, but any parameters set up remain in memory.

Output
Enter 2 digital output Channel numbers: (lo, hi).
On low alarm 'lo' goes from logic 0 to 1.
On high alarm 'hi' goes from logic 0 to 1.
For Match/Mismatch alarms enter the number as n,n.
Enter analogue output Channel (Analogue Output Module).

Text

Enter text number (1-40), for text to be output with logged (non-alarm) data.
Enter alarm text number (1-40), for text to be output only with alarm data or events.

If Counter-Timer inputs are entered

Select item to be defined:

Sensor

Select measurement type:

Freq	Select external or internal gate: External: permits an external gating signal, of up to 600 seconds, to be used. Internal: 0.1, 1, 10 or 100 seconds. Gives the average frequency over the defined gate time.
Total	Gives the running total or final total of input pulses which occur while the gate is enabled.
Period	Select number of periods averaged: 1, 10, 100, 1000, 10,000. Gives the average period of the input signal.
Intvl	Gives the time between a positive transition and the next negative transition (or the time between a negative transition and the next positive transition).
Off	Counter-Timer functions off, but any parameters set-up remain in memory.
Conv Alarm Output Text	As for Digital Input/Output Channels, pages 9.9 and 9.10.

If Analogue Outputs are entered

Select item to be defined:

[Voltage Out] or [Current Out] indicates the setting for the particular Channel(s).

Value — Normally used for setting-up and test purposes.
Enter percentage (0 to 100%) to obtain a fixed output in the range:
4 to 20mA, if Current is selected
0 to 10V, if Voltage is selected
(0% = 4mA or 0V. 100% = 20mA or 10V).

When VALUE is selected, the present output level is indicated, irrespective of whether it was set using VALUE, or by a measurement of the associated input Channel using the SCALE facility.

Scale — Enter value† for 0% current or voltage output.
Enter value† for 100% current or voltage output.

† The value relates to the measured result from the associated input Channel. Enter the numeric value only, the engineering units (e.g. μstrain, °C) will be automatically assumed depending on type of sensor connected.

Whenever the associated input Channel is scanned or monitored, the results obtained will caused the analogue output Channel to change to a proportional output level. This output level will be maintained until updated by the next scan or monitor measurement.

Enter task number: 1 to 8 (8 = highest priority, 1 = lowest priority)

A high priority task will interrupt a coincident lower priority task. To avoid lower priority tasks being locked-out, do not use high priority tasks for continuous scanning. Avoid continuous scanning of digital inputs.

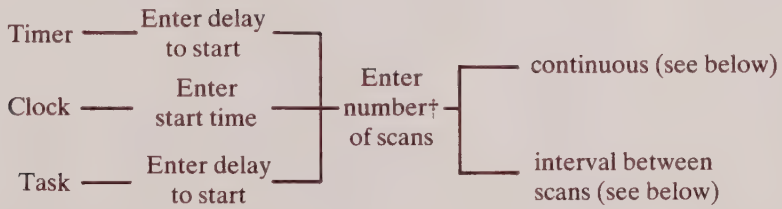
Enter task title: 18 characters maximum

Select task function:

Scan	For analogue, status and counter/timer inputs, and event inputs to be counted or totalised (see below).
Event	For detection of event inputs (see page 9.13).
Off	Turns off task, but leaves parameters in memory.

If Scan is selected

Select task trigger source:



Clock and Timer are derived from the internal clock. If "Task" is selected, the following example shows how the cross-triggering is implemented. Assuming that the Scan Task is Task 3 and that it is to be triggered from Task 4, then Task 4 must include the command to trigger Task 3. This command could result from e.g. an Alarm or a processing decision.

†If number of scans is '1', set 'continuous or fixed interval scans' to 'continuous'. The maximum number of scans which may be specified is 65535. To select an unlimited number of scans, press ⊡.
Event counting: period is defined by Scan Interval.

Continuous Continuous triggering allows the system to proceed efficiently by initiating the next scan when the last end-of-scan marker has been passed to the particular interface. The system is slowed down to the speed of the interface, thus preventing the measurement buffer becoming completely filled.

Interval between scans Interval triggering will initiate measurement at expected intervals subject to task priority. If the measurement buffer becomes completely filled, the system will wait for sufficient memory being available before continuing.

Enter channels to be scanned:

Enter channels included in this task. See rules on page 9.5.

To scan any Channel (*n*) *x* times, enter *n*, *n*, *n*, etc (*x* times), or to scan any Channel group (*m-n*) *x* times, enter *m-n*, *m-n*, *m-n*, etc (*x* times).

Channels can be entered in any numerical order, but will be scanned in the sequence in which they are entered. A group of Channels (*m-n*) can be scanned in numerical order (*m-n*), or reverse order (*n-m*), (e.g. 1–20 or 20–1).

For External Module Measurement, using the Digital Input/ Output Module, enter module numbers.

Select measurement rate

500 or 150/s*, 100/s, 40/s, 10/s
For analogue Channels only
*500/s if using Solid State Selector
*150/s if using Reed Selector
The measurement rate affects the scale length; this applies to ac inputs, which are scanned at 5/s.

Select end-of-scan processing

Off: no processing is applied
On: data in this Task is processed according to programs written as described in Chapter 12.

Select data to be logged:

All	Log everything
Alarm	Log only Channels in alarm
Proc	Log only processed Channels
Al + Pr	Log both
None	Log nothing

Select when to output task header:

Always	Output headers each time task is actioned
Log Data	Output headers only when data is logged

Select output device:	Tape	To built-in cartridge
	Print	To built-in printer
	RS232	To PL1 on Controller
	422/a	To PL1 on Serial Interface (if fitted)
	Dsply	To built-in display
	GP-IB	To PL1 on GP-IB Interface (if fitted)
Select format required:	Coded	Coded format is not offered for Printer or Display
	Full	
	Compact	

Enter tasks triggered on alarm [x,y,z – –]

Enter the numbers of any Tasks to be triggered if an Alarm, is detected in the current task.

0 = no trigger

If a task receives a cross-trigger while it is still active, the trigger will be ignored.

Enter tasks aborted on alarm [x,y,z – –]

Enter the number of any Tasks to be aborted if an Alarm is detected in the current Task.

0 is entered if no Tasks are to be aborted.

Ready for command

The Task menu is now complete, and ORION is ready for the next command.

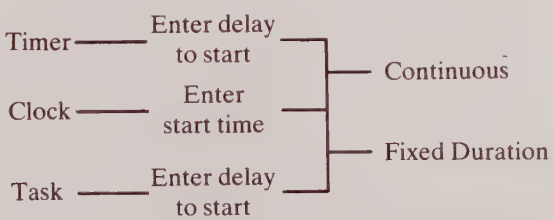
If Event is selected

Enter event channel numbers

Enter Event Channel numbers. See page 9.5.

Select what is to enable events

Starts the time 'window' in which events will be detected.



Clock and Timer are derived from the internal clock. If "enable Event from Task" is selected, the following example shows how the cross-triggering is implemented. Assuming that the Event Task is Task 3 and that it is to be enabled by a trigger from Task 4, then Task 4 must include the command to trigger Task 3. This command could result from e.g. an Alarm or a processing decision.

Select event data to be output:	All ("Select output device" next) None ("Enter tasks triggered etc." next)	
Select output device	Tape	To built-in cartridge
	Print	To built-in printer
	RS232	To PL1 on Controller
	422/a	To PL1 on Serial Interface (if fitted)
	Dsply	To built-in display
	GPIB	To PL1 on GP-IB Interface (if fitted)
Select format required:	Coded Full Compact	Coded format is not offered for Printer or Display

Enter tasks triggered on alarm [x,y,z,...]

Enter the numbers of any Tasks to be triggered if an Alarm, i.e. the arrival of an Event, is detected in the current task.
0 = no trigger
If a task receives a cross-trigger while it is still active, the trigger will be ignored.

Enter tasks aborted on alarm [x,y,z...]

Enter the numbers of any Tasks to be aborted if an Alarm, i.e. the arrival of an Event, is detected in the current task.
0 is entered if no tasks are to be aborted.

Ready for command

The Task menu is now complete, and ORION is ready for the next command.

Conversion Number: Enter 1 to 40

Conversion Type:	Scale	Enter m,c and units for 'mx+c units' (scaling and offset)
	Recip	Enter a and units for '$\frac{a}{x}$ units'
	Polyn	Enter a, b, c, d and units for '$ax^3 + bx^2 + cx + d$ units'
	Root	Enter a, b, c, d and units for '$(\sqrt{ax + b}\ ) \times c + d$ units'
		x in all expressions is the measured value.
	Off	De-selects the numbered Conversion, but leaves parameters in memory.

A Conversion is applied to specified input channels using the Channel Definition.

Enter text number [1-40]

Enter text (up to 18 characters)

Text is applied to specified channels using the Channel Definition. Any text (1-40) can be output with normal (non-alarm) logged data or with alarm-only data.

Facility:	Chans	Enter Channel Numbers of the analogue input Channels to be initialised; this measures and stores 'unstrained' parameters from strain gauge Channels. (Incidentally, it also sets-up auto-ranging; this is not essential for all Channels. See page 9.5.
		If a program is saved on the cartridge after initialisation, the initialised parameters (such as unstrained gauge values) will also be saved as part of the program.
	Refs	Enter Reference Junction Number (1-4) and enter Junction Temperature; this sets up a special 'cold-junction' reference for use with thermocouple Channels.
	Clock	Enter current Date and Time; this sets up the internal clock. The clock keeps track of year (including leap year) but indication of years is not output to the Display or recording device.
		When interrogating the clock, use ↓ to advance to the next function; do not use ENTER in this case as this will reset the seconds to zero (00), thereby causing small apparent time errors.
	Print	Normal = top reading is most recent. Inverted = bottom reading is most recent.
	EDIT	Enter program line number to begin edit. Each time that ENTER is pressed, the next line number will be presented.
	EXIT	To retrieve "Ready for command" state after editing.

Select tape handling required

Progr	Refers to total Channel and Task databases.
Data	Refers to logged data (results)
Direc	List of numbers and names of programs.
Mark	Reserves spaces for programs (MARKer).
Rewind	

If Progr is selected Select program facility required

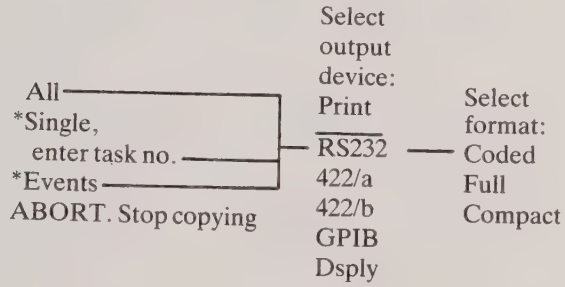
Load — Enter program number [1-7]
The specified program is loaded from the cartridge into memory

Save — Enter program title [40 chars max]
Enter program number [1-7].
The specified program is output from memory and saved on the cartridge.
For strain gauge channels, any initialised values will also be saved.

If Data is selected Select data handling required

Copy — Causes the specified data to be copied from the cartridge to an external output device.

Select task data to be copied



*For use with cartridges recorded in CODED format only.

Mode — Normal: Write data until cartridge is full.
Cyclic: Write data until cartridge is full, then start to overwrite.

Erase — Logged data erased, but not programs.

If Direc is selected Select output device

Print — Built-in printer
RS232 — PL1 on Controller

'Directory' is a list of the numbers and names of the Programs already stored on the cartridge; the list is output to the specified output device.

If Mark is selected

Enter number of programs to be reserved (maximum on one cartridge is 7). Note that one program uses 6% of tape capacity.

If Rewind is selected

Tape rewinds to start. It is advisable to rewind the tape before removal and storage of cartridge, this ensures correct starting position for tape when cartridge is subsequently used again.

Item:	Chan	For Channel Definitions (1-200)*
	Task	For Task Definitions (1-8)*
	Conv	For Conversion Definitions (1-40)*
	Text	For Text Definitions (1-40)*
	Progr	For processing programs
	ABORT	Stops listing

*Define m or m-n, where m = single definition,
m-n = consecutively numbered group of definitions

Output Device:	Displ	To built-in display
	RS232	To PL1 on Controller
	Print	To built-in printer

Press to advance printer paper; hold for auto repeat.

3.3 EDIT KEYS

The EDIT keys act upon the information shown in the Display as described below. All eight keys are available for use when writing or editing processing programs, and all except LINE INSERT can be used when setting up the various menus.

All the keys have an auto-repeat facility. If pressed and released quickly, only one step is performed, e.g. one character is deleted. However, if the key is held pressed, the relevant action will be repeated, e.g. characters will continue to be deleted, until either the key is released, or there are no characters left in the line.

To edit a processing program which is not currently being displayed, first press INIT, followed by the soft key EDIT. Enter the number of the line which is to be edited, then use the EDIT keys as described in the following sections.

On completion of editing, pressing INIT followed by the soft key EXIT will produce the displayed state "Ready for command". Alternatively, the user may simply press the key for the next required function, e.g. RUN.

STEP BACK

STEP FORWARD

These keys enable the user to step through processing programs line by line in either direction, for inspection and/or alteration. Branches of menus can be similarly explored in most cases. If the sequence under investigation is already displaying the final permissible line, in either direction, any attempt to step further in that direction will cause the Error Tone to sound, as described in Section 10.

STEP LEFT

STEP RIGHT

These keys enable individual characters in the Display to be altered as follows. The ⇐ key must be pressed first. This causes the cursor, in the form of a blinking rectangle, to appear superimposed on the extreme right-hand character in the line. Further operation of this key, plus the ⇒ key if necessary, will align the cursor on top of the character to be changed.

When the new character key is pressed, the cursor will move one step to the right, thus revealing the new character in place in the Display. There is no need to move the cursor clear of the Display before pressing ENTER.

CHARacter INSERT

To insert a character into the information in the Display, first press ⇐ to produce the cursor. Position the cursor on the character which will *follow* the one to be inserted, and press CHAR INSERT. A gap will appear in the displayed information, in which the new character can be inserted.

CHARacter DELETE

To delete a character, first press ⇐ to produce the cursor. Position the cursor on the superfluous character, press CHAR DELETE, the character will disappear and the remaining text close up to fill the gap.

LINE INSERT

This facility is available for use only on processing programs. See the EDIT section in Chapter 12.

LINE DELETE

To delete a line, operate the ↑ or the ↓ keys until the line to be deleted appears in the Display. Press LINE DELETE, the original line will disappear and the remaining lines automatically close up so that the next one in the sequence now appears in the Display. With processing programs, the line numbering is automatically adjusted.

4 KEYPAD

The Keypad contains all the alpha-numeric characters together with other symbols and functions. Most of the keys have a dual purpose, the SHIFT key being used to select the required character or function.

Fig 9.2 *The Keypad*

The alpha-numeric keys are self-explanatory, but certain specialised keys are described below.

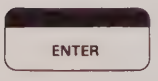
Selects the active function of all the dual function keys, including the eight PROCESSING keys.

Press SHIFT key once to select SHIFT, i.e. the alpha characters and the titles above the PROCESSING keys are active (SHIFT-key annunciator illuminated).

Press SHIFT key again to cancel SHIFT, i.e. the numeric characters, or symbols, and the titles on the Processing keys are active. (SHIFT-key annunciator not illuminated); this is the condition which the logger assumes at switch-on.

e.g. Alpha character: active when SHIFT is selected → ← Numeric (or other) character: active when SHIFT is cancelled

Enters a space between characters.



Enters any selected character(s) into memory.

This key can be used for one of three functions:

1. Selects 'an unlimited number of scans' (Task Definition).
2. Selects a 'dont-care' mask on any bit(s) in an alarm bit pattern (Channel Definition)
3. Enters a [*] for use in multiplication or exponential expressions, e.g. an expression such as 5×3 can be entered as 5 [*] 3; an expression such as 5^3 can be entered as 5 [*] [*] 3 (see NB below).

NB. The [*] [+] [-] and [/] keys, are intended solely for use in entering character strings as part of a program. The keypad cannot by used (like a calculator) to perform mathematical functions.

For use when entering leading spaces in a message: the spaces + message are enclosed by two back-slashes.

For example, to enter the word NAME with 4 leading spaces, press the following keys.

This will appear on the display thus:

Selects the 'text tone', to be output with an associated message (see Audible Indications, Section 10).

When entered as part of a message, the logger will output the text tone each time the message is output or recalled.
For example, the tone may be included in the following alarm message.

Whenever the alarm occurs, the message will be output and the tone will be heard.

Note that, when a message containing the [♪] character is output to an external device (via one of the logger Interfaces) the [♪] character is sent as 'bel' in ASCII. 'Bel' is a non-printing character which will cause an audible tone at the device.

5 SOFT KEYS

Each Soft Key has a meaning which varies depending on the Function selected. The current meaning of each Soft Key is indicated by the word displayed immediately above the key on the lower line of the Display; this permits one of the items offered (up to 6) to be chosen. When a Soft Key is pressed the function displayed above the key is accepted into memory and the program jumps to the next line. The Display may offer a further selection of functions or alternatively may request entry of other information using the keypad. This process continues until the program reaches the end. If it is necessary to gain access quickly to a particular line of program, such as when making minor changes to an existing program, the ↑ ↓ keys can be used to make the program roll forward or back without making any entry of parameters.

When a program line is reached which contains a soft key choice, the function which was previously selected will be flashing on the Display. This indicates the current selection, it may be altered by pressing a different soft key.

6 KEYSWITCH

OFF — Logger off, except for CMOS clock which keeps track of date and time, and is maintained by a battery for up to 100 days. The program, contained in the internal memory, is lost when the OFF position is selected.

STANDBY — As 'OFF', except that the current program is maintained in memory for at least 6 hours.

ON — Logger on and fully operational.

LOCK — As 'ON', except that Front Panel controls can be used to examine the program but not to change it. (MON and LIST keys function normally),

7 DISPLAY

The display is a 2-line device which can accommodate 40 characters on each line. The display has two prime functions:

1. To show prompt messages and soft-key options when programming the logger.

2. To show measured results from logging routines, or when monitoring (MON), and to show program listings (LIST)

Displayed Symbols

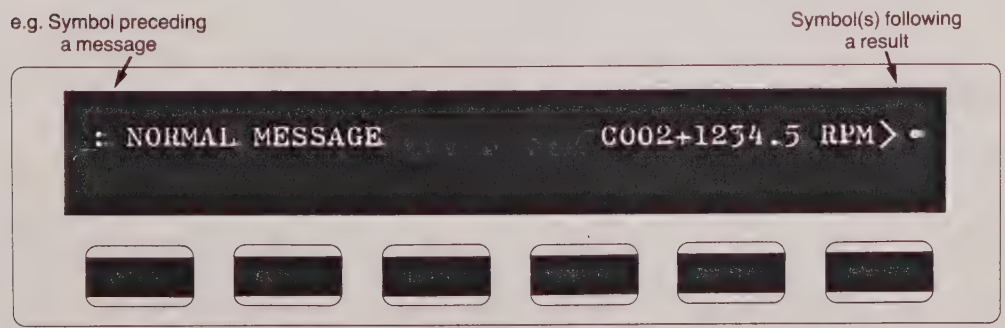
Displayed messages and results will be accompanied by certain symbols when appropriate; the symbols indicate the status of the message or result. The symbols, which are generated automatically by the logger, are described below.

Symbols preceding a message

: = normal message
! = alarm message

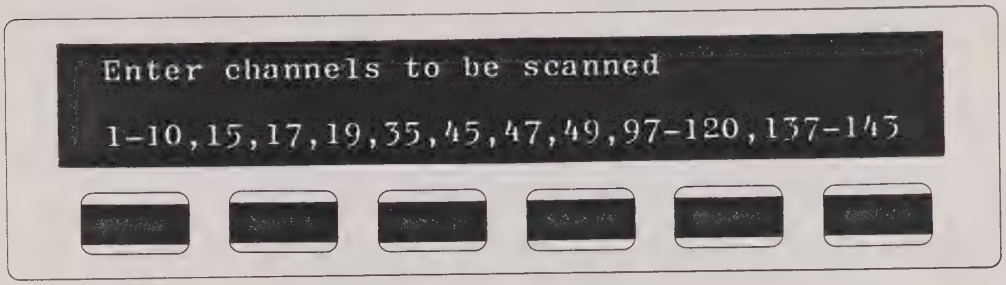
Symbols following a result

First column indicates Alarm status
Second column indicates Processing status.
See Chapter 11, page 11.12.



8 SCROLLING FACILITY

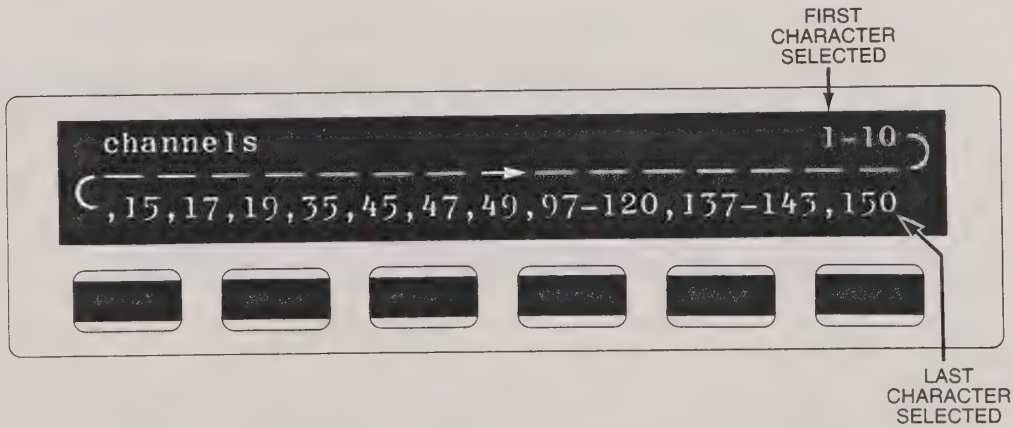
An automatic scrolling facility operates when Channel numbers are being entered, both top and bottom lines of the Display being used. When selected, the Channel numbers firstly appear on the bottom line of the Display (in the normal way), until the bottom line is full.



e.g. Entering channel numbers

The bottom line will accept up to 40 characters. When the 41st character is selected the scrolling facility automatically operates. The scrolling facility causes the prompt message to be condensed to one word (i.e. 'channels'), and the remainder of the top line is then available to accommodate Channel numbers.

As each subsequent character is selected, it appears at the right hand end of the bottom line, and all previously entered characters move to the left one place. The character at the left hand end of the bottom line moves up to the right-hand end of the top line.



In this way, both lines of the Display are used to accommodate Channel numbers, up to a maximum of 71 characters.

9 POWER ANNUNCIATORS

ON	Illuminates when the ac mains supply is sufficiently high to run the logger (mains switch at rear must be set to ON)
EXT dc	Illuminates when the external dc power source is sufficiently high to run the logger (when a battery is connected to the DC SUPPLY terminals; not affected by mains switch).

back-up — Illuminates when the internal rechargeable back-up battery holds sufficient charge to maintain the internal memory (which contains the program).

All 3 annunciators may be operational when the keyswitch is set to ON, STANDBY or LOCK. None of the annunciators are operational when Keyswitch is set to OFF.

If the ON annunciator is illuminated but EXT dc is not, then the logger is being powered from the ac mains supply. If the EXT dc indicator is illuminated but ON is not, then the logger is being powered from the external dc source. If both annunciators are illuminated either power source may be supplying the power. The back-up annunciators should be illuminated at all times when either or both ON and EXT dc are illuminated, and for up to 6 hours after both power sources have failed. The back-up battery, when fully charged, will maintain the internal memory for a maximum of 6 hours. When the back-up annunciator is no longer illuminated the charge in the back-up battery is insufficient to maintain the memory, causing the contents of the memory to be lost. Memory is also lost when the keyswitch is set of OFF, regardless of the state of charge in the back-up battery.

Summary of normal annunciator indications.

KEYSWITCH SETTING	POWER ANNUNCIATORS		
	ON	EXT dc	back-up
OFF	off	off	off
ON	on/off*	on/off*	on
STANDBY	on/off*	on/off*	on
LOCK	on/off*	on/off*	on

*depends on power source(s) connected.

10 AUDIBLE INDICATIONS

A small built-in loudspeaker, located behind the front panel, gives two types of audible indication.

1. Acknowledge tone: this is a short 'beep' which is produced whenever any front panel key is pressed and the function or character selected is valid; the acknowledge tone indicates that the function or character selected has been accepted by the logger.

2a. Error tone: this is a series of interrupted 'beeps' which is produced whenever any front panel key is pressed for which the function or character selected is invalid; the error tone indicates that the function or character has not been accepted by the logger. (No rectification action is necessary, simply select the correct key).

2b. Text tone: as for error tone, but produced as a result of a text tone ♪ being specified as part of a text message; the text tone will be heard whenever the particular text message is output or recalled. The text-tone can be applied to any text message(s), it may be particularly useful if applied to an alarm text.

Chapter 10
Data Reduction Facilities

Contents

		Page No
1	Introduction	10.2
2	Conversions	10.3
3	Alarms	10.5
3.1	Examples	10.8

1 INTRODUCTION

ORION provides a comprehensive selection of data reduction facilities which are classified as follows: Conversions, Alarms and Processing. Conversions and Alarms are described in this Chapter, Processing is explained in Chapter 12.

Any one, or all, of the data reduction facilities can be applied to any input channel or group of channels.

If Conversion and Alarm, or Conversion and Processing, are applied to the same channel, Conversion operates on the measured results first. The values and units obtained from the Conversion are then used in the Alarm or Processing functions.

A log can be requested of readings in alarm, or of processed results, or both.

Conversion and alarms are linked to the channel (using Channel Definition). If a channel is included in several tasks then any of these tasks can detect or reset the alarm, or output processed data, for that channel.

Numeric entries Conversion constants, and Alarm levels, may be entered as ordinary numbers when they are within the range 10^{-7}. Numbers outside this range, up to the maximum of 10^{-15} to 10^{15}, should be entered in exponential form. For example, to enter 37.5 million, enter 37.5 EXP 6. This will be displayed as 37.5 E6 (E6 implies $\times 10^6$).

2 CONVERSIONS

The measurements are mathematically converted to user-defined units; any one of four different types of conversion can be selected:

Scaling and offset (linear)
Polynomial
Reciprocal
Square Root

It is the converted results that are used in any subsequent Alarm detection or Processing action.

Conversions are already built in for certain specific types of input. For example, the results from any thermocouple or PRT channel are automatically output in °C, the results from any strain channel are automatically output in μstrain. If required, additional conversions can be applied to these or any other inputs, using Conversion Definition.

The Conversion Definition facility permits the user to apply one of four types of mathematical conversion to any measured value, and thereby converts the value to specified engineering units. The four types of Conversion are: Scale (Linear), Polynomial, Reciprocal and Square Root. Up to 40 Conversion equations of mixed type can be stored. Each equation has a Conversion number (1 to 40), and is assigned to a particular channel (or group of channels) by entering the number in the appropriate part of the Channel Definition.

Numeric entry is limited to the first six significant figures. If the number entered is greater than 524,288 or less than 0.0039 then, when that number is recalled it will be displayed in exponential form with 3 significant figures, but the whole original number (up to six figures) will be retained in memory and used in the calculations.

The four types of Conversion equations are described below.

Scale This gives a linear, scaling and offset, conversion. Enter the constants m and c, and units for the Scale equation:

$$y = m\,x + c\ units$$

where m = scaling factor
c = offset
units = any characters (3 max)
x = measured value
y = Conversion result

e.g. The Scale equation may be used to convert °C to °F. The logger measures temperature in °C, and so by applying the equation ($°F = \frac{9}{5}\,°C + 32$) to the measured value, the temperature in °F is obtained. This is achieved as follows:

$m = \frac{9}{5}\ (= 1.8)$
$c = 32$
units = FAR (degrees Fahrenheit)

The Scale equation '$y = m\,x + c\ units$' thus becomes:

$$y = (1.8\,°C + 32)\,FAR$$

If the measured temperature was 15°C, the result would be:

$$(1.8 \times 15) + 32 = 59\,FAR$$

The units in the above example are entered as 'FAR', this being an abbreviation for °F. All results from this Conversion will include 'FAR', thereby giving indication of the correct units. Depending on the application, any suitable units can be entered.

If required, units only can be changed without any mathematical conversion being applied. This may be achieved using a Scale Conversion, and setting m to 1, c to 0 and units as required.

When a Conversion is applied to any channel, the units which would normally be output in the results (eg V, °C, μstrain, etc) will be cancelled, even if no units are included in the Conversion.

Polynomial This gives a cubic conversion of the form $y = ax^3 + bx^2 + cx + d$ units

where a, b, c and d are constants
units = any characters (3 max)
x = measured value
y = Conversion result

Enter a, b, c, d and units to obtain a cubic conversion. To obtain a square-law conversion set a to 0.

Reciprocal This gives a fractional conversion of the form $y = \frac{a}{x}$ *units*

where a is a constant
x = measured value
units = any characters (3 max)
y = Conversion result

The Reciprocal conversion could typically be used in a stress/strain calculation such as $E = \frac{stress}{strain}$, where strain (in μe) is the measurement from a strain gauge channel.

Root Provides a square root conversion of the form $y = (\sqrt{ax + b}) \times c + d$ *units*

where a, b, c and d are constants
x = measured value
units = any characters (3 max)
y = Conversion result

The Root conversion could typically be used to calculate flow rate, using the results from a differential pressure transducer. Alternatively, just the square root part of the equation can be used on its own ($\sqrt{x}$) by setting a to 1,
b to 0,
c to 1,
and d to 0.

For values of x which yield a negative result for the equation $(ax + b)$, the results will be invalid since the square root will be evaluated by negating the −ve sign.

Units In the absence of a Conversion being applied, all measurements output by the logger are accompanied by their respective units, ie V, Ω, mA, μ strain, or °C. The Conversion Definition facility permits the user to apply different engineering units, as may be required, e.g. PSI, RPM, CMS, etc. When a Conversion Definition is applied to any measurements, the normal units output by the logger are cancelled (even if no units are specified in the Conversion). Units specified in a Conversion can consist of any characters, up to a maximum of 3.

If it is desired to enter units only, use a Scale Conversion, setting m to 1 and c to 0.

3 ALARMS

Measurements, or converted results (if Conversions are applied), are compared with pre-defined levels. When the levels are exceeded an alarm state is detected, and any one, or all, of the alarm actions occur:

Results logged
Digital outputs activated
Another task is triggered

Four different types of alarms state can be defined:

Limits
↑ Threshold
↓ Threshold
Change

The converted results from the input channels are compared with the alarm values. If the results are found to be in an alarm state, one or all of the following alarm actions will occur (as selected):

Primary alarm actions

1. ALARM results only are logged (selected in Task Definition, Data to be logged).

2. ALARM text is output (selected in Channel Definition, Enter alarm text number; text is defined in Text Definition).

3. One, or two, digital output channels produce a change of logic state (selected in Channel Definition, Alarm Outputs). (See also page 10.7).

Secondary alarm actions

4. Task will Continue, Abort or Pause (selected in Task Definition, Action on Alarm).

5. Another task is triggered (selected in Task Definition, Trigger other tasks).

Alarm states can be specified for any input channels. For analogue inputs, and counting and digital bcd inputs, one of four types of alarm can be selected:

Limits
↑ Threshold
↓ Threshold
Change

} Each type is described below under a separate heading

For digital status inputs, alarm states of Pattern Match or Pattern Mismatch can be specified, see Chapter 5, page 5.4. The alarm states for all inputs are selected using Channel Definition. Any 'event' detected on a digital event input is automatically treated as an alarm.

Limits (Fig 10.1) High and low limits are entered. An alarm occurs if the measured result is > high value, or < low value.

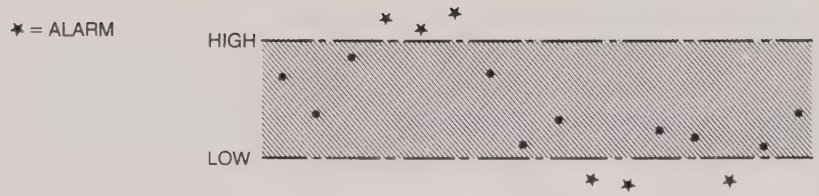


Fig. 10.1 *Alarm: High and Low Limits*

↑ **Threshold** (Fig. 10.2) A high limit sets the 'threshold', and a low limit defines a 'dead band' below the threshold. The first reading above the threshold gives an alarm. Thereafter an alarm occurs only when readings cross the entire band in a positive (↑) direction.

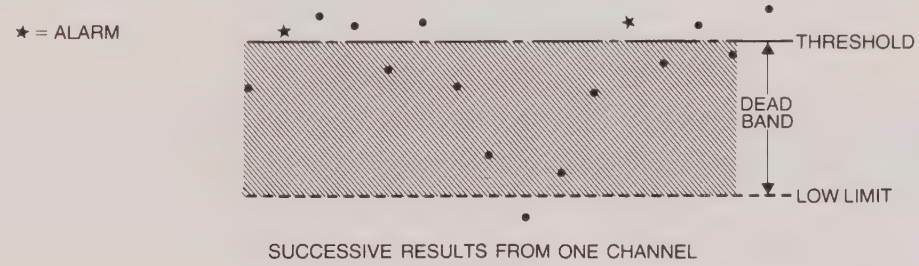


Fig. 10.2 *Alarm: ↑ Threshold*

↓ **Threshold** (Fig. 10.3) A low limit sets the 'threshold', and a high limit defines a 'dead band' above the threshold. The first reading below the threshold gives an alarm. Thereafter an alarm occurs only when readings cross the entire band in a negative (↓) direction.

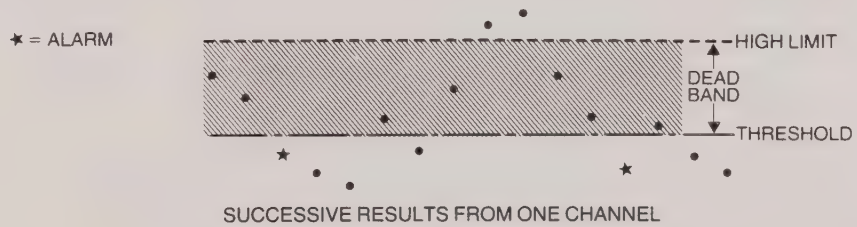


Fig. 10.3 *Alarm: ↓ Threshold*

Change (Fig. 10.4) A value of change is entered. The first result always constitutes an alarm, and the value of this result becomes the datum level for observing changes. Thereafter, an alarm occurs only when the result lies outside the datum ± change levels. When an alarm occurs, the datum moves to the value of the alarm result, and subsequent readings are compared with the new datum ± change. Each subsequent alarm causes the datum to adopt the new alarm value.

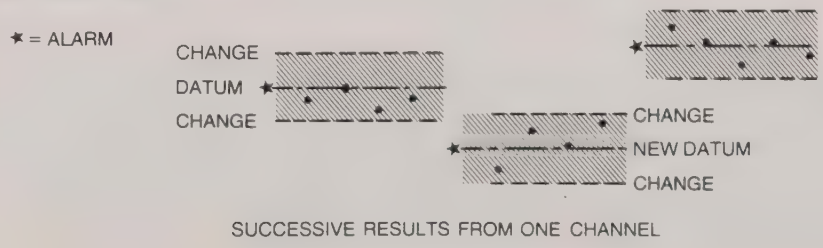


Fig. 10.4 *Alarm: Change*

The ↑ and ↓ Threshold alarms, which incorporate a dead band, give a hysteresis such that results are logged only when readings have crossed the specified band. In contrast, the Change alarm gives notification that readings have moved up or down by more than a predetermined amount.

Alarm Outputs Any digital output channel, contained on a Digital Input/Output Module 35302B, can be used as an alarm output as described in Chapter 6. One, or two, alarm outputs can be assigned to an input channel or group of channels. These two alarm channels are referred to as 'lo, hi'. The lo channel changes logic level when the result from an input measurement goes lower than a pre-determined limit; the hi channel changes logic level when the result from an input measurement goes higher than a pre-determined limit. The alarm outputs can thus give indication that the input to which they are assigned has gone out of high or low limits.

The output Channels are at Logic 0 in the quiescent, i.e. non-alarm state. The actual voltage levels of Logic 0 and Logic 1 are defined in the Specification at the end of Chapter 6. When an alarm is detected, the relevant alarm output Channel will change state from Logic 0 to Logic 1. Logic 1 represents the conducting state of the output switch, i.e. switch on.

The output will remain at Logic 1 until the next non-alarm measurement is detected.

The table below summarises alarm output action.

ALARM STATE	ALARM OUTPUTS	
	lo channel	hi channel
When < Low alarm state is detected	changes to Logic 1	
When > High alarm state is detected		changes to Logic 1

The lo/hi alarm channels which may be specified for each type of alarm, and the resulting action of the alarm outputs, is shown in the table below.

	ALARM TYPE	ALARM CHANNELS WHICH MAY BE SPECIFIED	ACTION OF ALARM OUTPUTS
Applies to Analogue inputs, and Digital Counting and BCD inputs only	High and Low Limit	hi and/or lo	hi channel goes to Logic 1 when High Limit is exceeded, lo channel goes to Logic 1 when Low Limit is exceeded.
	↑ Threshold	hi only	hi channel goes to Logic 1 when entire dead band is crossed in a positive direction.
	↓ Threshold	lo only	lo channel goes to Logic 1 when entire dead band is crossed in a negative direction.
	Change	hi and lo	hi channel goes to Logic 1 when a more positive Change is detected, lo channel goes to Logic 1 when a more negative Change is detected.
Applies to Digital Bit inputs only	Match/Mismatch	hi or lo	hi and lo channel go to Logic 1 when a Match or Mismatch is detected.
Applies to Digital Event inputs only	Event (lo-to-hi transition)	hi only	hi channel goes to Logic 1 when a lo-to-hi Event transition occurs
	Event (hi-to-lo transition)	lo only	lo channel goes to Logic 1 when a hi-to-lo Event transition occurs
	Event (Any transition)	hi and lo	hi channel goes to Logic 1 when a lo-to-hi Event transition occurs, lo channel goes to Logic 1 when a hi-to-lo Event transition occurs.

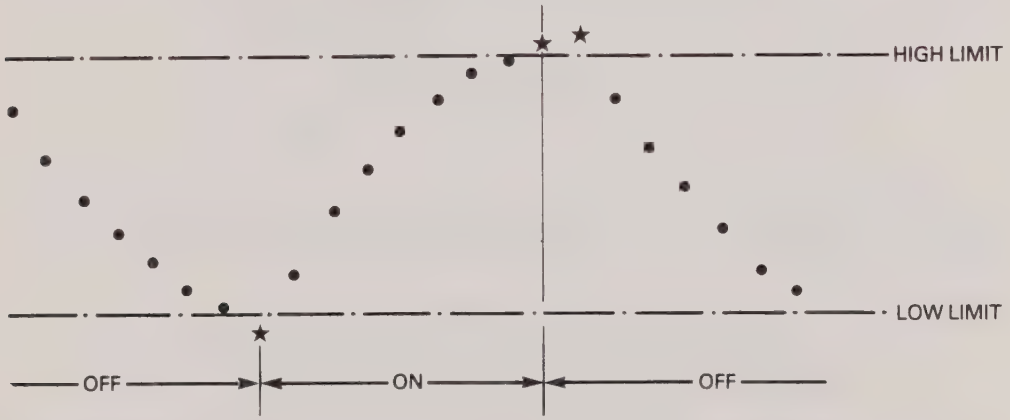
The numbers of the alarm output channels (hi and/or lo) are defined in the 'Alarm Outputs' part of a Channel Definition. The one, or two, channel numbers are entered as 'lo, hi'. For example to make channel 41 the lo channel and channel 42 the hi channel, enter '41, 42'. To select the lo channel only, enter 'n, 0'; to select the hi channel only, enter '0, n',

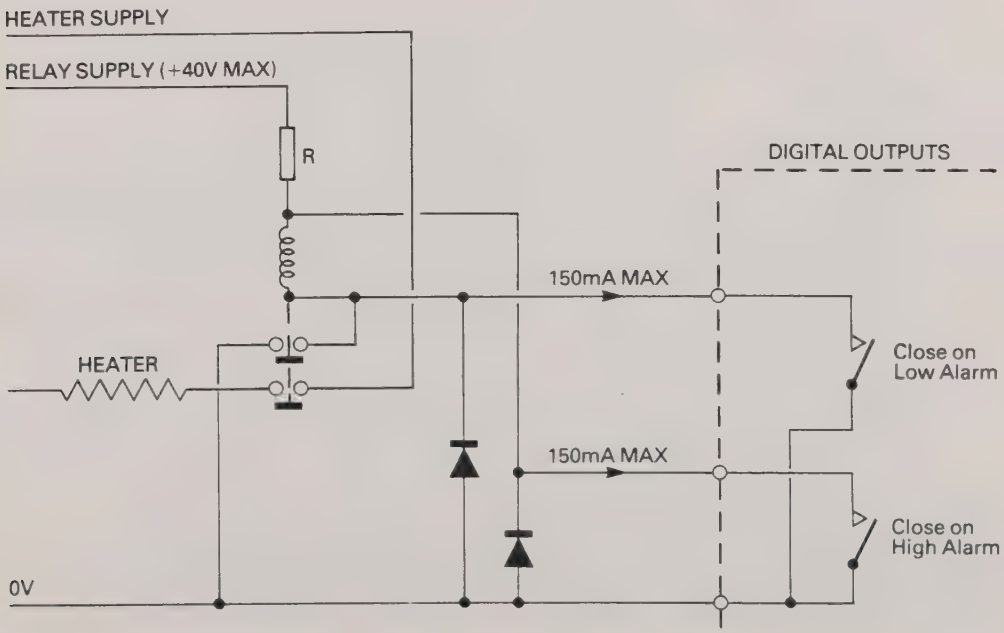
where n is the channel number required. The same channel may be used to indicate both High and Low states, this is achieved by entering 'n, n'; channel n will then be activated by either a High or Low alarm.

3.1 Examples

ORION is able to provide outputs which control external equipment – essentially to switch devices on or off. In simplest concept this is fulfilled by an Alarm output that can cause a lamp to light, or perhaps switch on a heater when a temperature reaches a pre-defined low level. Thus by arranging outputs that are associated with measurements (or events) on specific channels an elementary control arrangement may be envisaged.

Many devices may require a supply which cannot be switched directly by ORION – a heater operated from the ac mains is an example. It is then necessary to interpose a relay which can be switched by ORION, the relay contacts carrying the prime load. The example below illustrates this sort of application in which a heater (for an oil bath perhaps) comes on at a low temperature and goes off at a high temperature. A measurement channel repeatedly measures the oil temperature and compares each result with High and Low Alarms. Two digital output channels are associated with the measurement channels, one to operate on the Low Alarm, the other on the High Alarm.





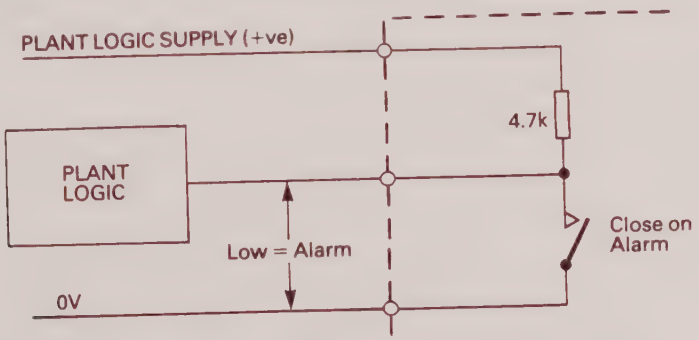
The digital outputs which are shown here as simple switches are actually field-effect transistors. Internal protection exists for excessive positive voltage, but it is desirable to include the two diodes shown to ensure that negative transients cannot harm the switches. The value of R depends on the voltage of the relay supply and must limit the switch current to <150 mA. The relay needs the hold-on contacts because the Low Alarm switch, having brought in the heater, will open again for the first temperature measurement that is above the low limit.

The use of the outputs as simple switches has the advantage that various devices, each with their own separate supply, can be controlled; the 0V return must be common to all devices for any one output module, but may float at up to ±100V from earth (logger chassis).

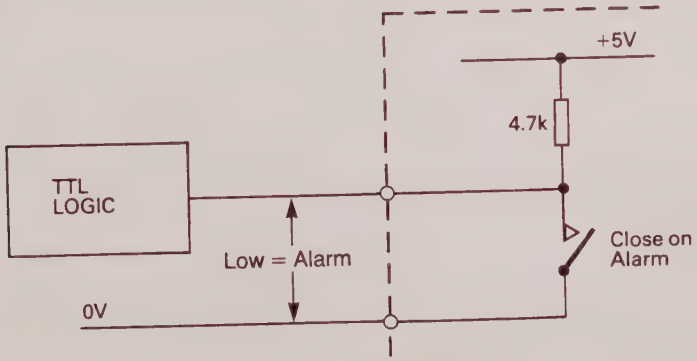
Logic outputs

Instead of using the outputs to give simple switching of external devices, they can be rearranged to give outputs of logic levels. The simplified versions shown below are achieved by links within the 35302B module and are described in detail in Chapter 6, Section 3.

When it is required to interface directly with plant logic operating at 24V (or 12V) the outputs switches can be powered from the logic supply. Although the maximum is specified as +25V (continuous), no difficulty should be experienced with conventional 24V logic which may be expected to have occasional excursions to 27V, or even 30V.



If the logic to be controlled is TTL, obviously the external supply could be +5V; but it may be more convenient to use the supply provided within the 35302B module. If external or the internal supply is used, all outputs in the one module are comitted to that mode of operation.



The use of alarm outputs in the ways outlined above provides external action that results directly from a measurement, a status or an event occurring at a specific channel. In the case of measurements, high or low limits, thresholds or a change can determine the conditions for which the output occurs. Thus, within one Task, alarm outputs associated with various channels can cause external action that depends on the results from those channels.

Chapter 11
Recording Devices and Output Formats

Contents

		Page No
1	Introduction	11.2
2	Printer	11.3
2.1	General Information	11.3
2.2	Print-Out Formatting	11.4
3	Cartridge Recorder	11.7
4	Output Formats	11.11
4.1	Adding Text	11.11
4.2	General Details	11.12
4.3	Coded Format	11.17

1 INTRODUCTION

A paper-roll printer and a magnetic-tape cartridge recorder are built into ORION as standard. Additional external recorders may be connected to ORION using the standard RS232 interface (PL1 on Controller) or the optional GP-IB or Serial Interface.

The recording device, to which results are to be output, is selected using the Task Definition. All measurements can be logged at any one device, or alternatively, selective logging to different devices can be obtained. This could be achieved using two tasks which both scan the same channels; one task could log all normal measurements to one device (e.g. cartridge), the other task could log alarm-only results to a different device (e.g. printer).

In addition to selecting the required recording device, the user may choose the output Format (see page 11.11); Full, Compact or Coded format may be chosen. The content of any one type of format is identical regardless of the type of output device used. Text can be defined for inclusion in full format.

Output devices and formats for Monitoring are selected independently, and may be different from those chosen for Task measurements. Monitored data may be output to the Display, Printer or RS232 (PL1 on Controller) only.

2 PRINTER

2.1 General Information

The printer has a thermal print head giving 20 characters per line; each character is formed from 5 x 7 dot matrix. The printing speed is two lines per second for lines with 20 characters. The printer produces instant hard copy onto the 2½'' wide paper roll. Approximately 20,000 lines can be printed on one paper roll.

When Full format is requested the print-out includes text, such as task titles or messages that have been attached to specific channels, in addition to the numeric data, units and symbols. Full format uses two lines for each channel measurement. If it is important to save space, Compact format should be selected. In Compact format text is not included and each measurement occupies one line.

Using the INITIALISE (Printer) facility the print-out can be set to Normal or Inverted (see Chapter 9).

Normal

The top reading is the most recent. The results are directly readable but are in reverse chronological order.

e.g.

```
D T 1

C 005 1.999      PSI
 :SENSOR 5
C 004 2.000      PSI
 :SENSOR 4
C 003 1.999      PSI
 :SENSOR 3
C 002 2.000      PSI
 :SENSOR 2
C 001 2.002      PSI
 :SENSOR 1
S T 1 07:16:28.2
  SCAN CHANS 1-5
```

Inverted

The results are in chronological order, but they are presented upside down as paper comes out of printer. (Tear off paper, and invert to read results).

e.g.

```
  SCAN CHANS 1-5
S T 1 07:17:03.8
 :SENSOR 1
C 001 1.995      PSI
 :SENSOR 2
C 002 2.000      PSI
 :SENSOR 3
C 003 1.999      PSI
 :SENSOR 4
C 004 1.999      PSI
 :SENSOR 5
C 005 1.999      PSI

D T 1
```

The example print-outs above show a scan of channels 1-5 as presented in Normal and Inverted format.

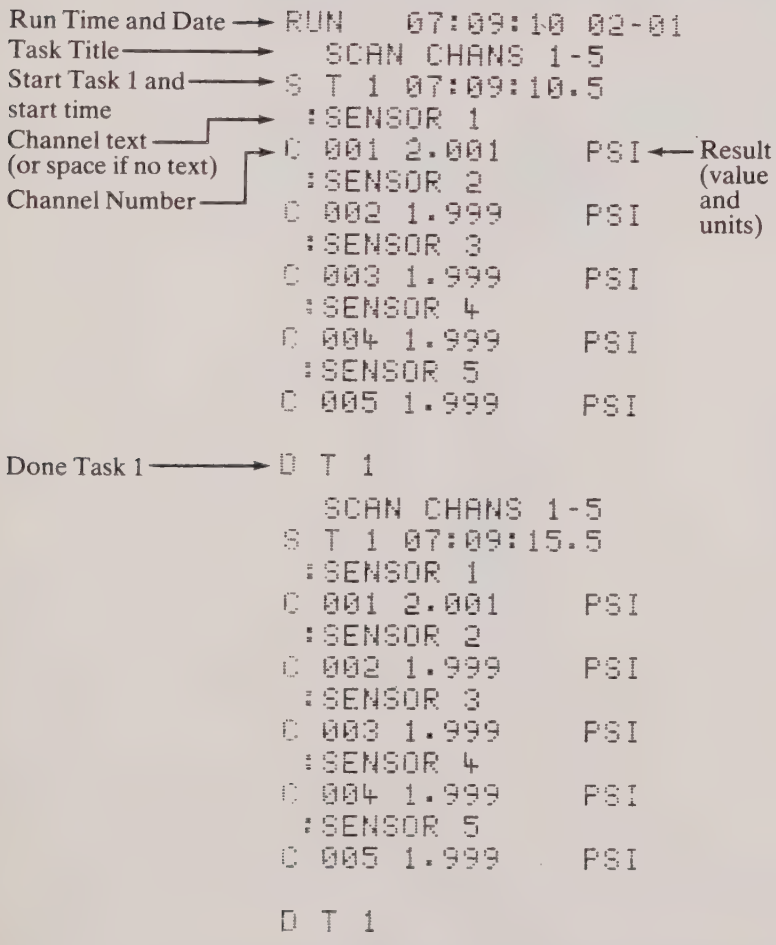
Instructions for changing the paper roll will be found on page 2.3. If the paper roll is to be changed when the logger is running, the PAUSE/CONTINUE facility should be used (this will stop output to the printer without affecting Tasks in progress, see page 2.5).

2.2 Print-out formatting

Detailed information on Output formats is given on pages 11.11-11.14, but to illustrate the print-out formatting, some simple example print-outs are shown on pages 11.4-11.7. The Full and Compact Format examples below show a scan of channels (1-5). A conversion has been applied to all channels, giving results in PSI. Each channel has an associated text applied, which has been used to indicate Sensor number. Note that, Formats without headers are available only by remote command (TAsk MArk NOne); see the associated publication 'Communicating with Orion *delta*'.

Full format

Full Format with headers



Full Format, without headers

i.e. Task title, and STn (Start Task n) and DTn (Done Task n) headers are omitted.

```
RUN   07:10:16 02-01
 :SENSOR 1
C 001 1.985     PSI
 :SENSOR 2
C 002 2.000     PSI
 :SENSOR 3
C 003 1.999     PSI
 :SENSOR 4
C 004 1.999     PSI
 :SENSOR 5
C 005 1.999     PSI
 :SENSOR 1
C 001 1.995     PSI
 :SENSOR 2
C 002 2.000     PSI
 :SENSOR 3
C 003 1.999     PSI
 :SENSOR 4
C 004 1.999     PSI
 :SENSOR 5
```

In Full Format, each result (which consists of a 40-character string) is presented on two lines, with 20 characters per line.

Compact Format

Compact Format, with headers

```
RUN    07:07:23 02-01
S T 1  07:07:23.7
C 001  1.991      PSI
C 002  2.000      PSI
C 003  1.999      PSI
C 004  2.000      PSI
C 005  1.999      PSI
D T 1
S T 1  07:07:28.7
C 001  2.002      PSI
C 002  1.999      PSI
C 003  1.999      PSI
C 004  2.000      PSI
C 005  1.999      PSI
D T 1
```

Compact Format, without headers

i.e. Task Title, and STn (Start Task n) and DTn (Done Task n) headers are omitted.

```
RUN    07:11:28 02-01
C 001  1.998      PSI
C 002  2.000      PSI
C 003  1.999      PSI
C 004  2.000      PSI
C 005  1.999      PSI
C 001  1.985      PSI
C 002  2.000      PSI
C 003  1.999      PSI
C 004  2.000      PSI
C 005  1.999      PSI
```

Alarms and Processing

The following example (in Full Format) shows Alarm and Processing information. Each channel has 2 texts applied:

non-alarm text = SENSOR n
alarm text = SENSOR n ALARM
where n is the Channel Number

This column gives Alarm indication and status (> = result greater than high limit, < = result less than low limit)

This column indicates when processing has occurred (• = processed, blank = unprocessed)

: = Non-Alarm Text

! = Alarm Text

```
S T 1  02:46:15.8
:SENSOR 1
C 001  01.9993   PSI
!SENSOR 2 ALARM
C 002  01.9993   PSI<
 !SENSOR 3 ALARM
C 003  01.9993   PSI>
 :SENSOR 4
C 004  7.99736   PSI •
 !SENSOR 5 ALARM
C 005  5.99809   PSI>•
 :MEAN
P 106  3.99869   PSI •
D T 1
```

• ← Processed result from Channel variable

>• ← As above, plus high alarm

• ← Processed result from program variable

See page 11.13 for all Alarm and Processing symbols.

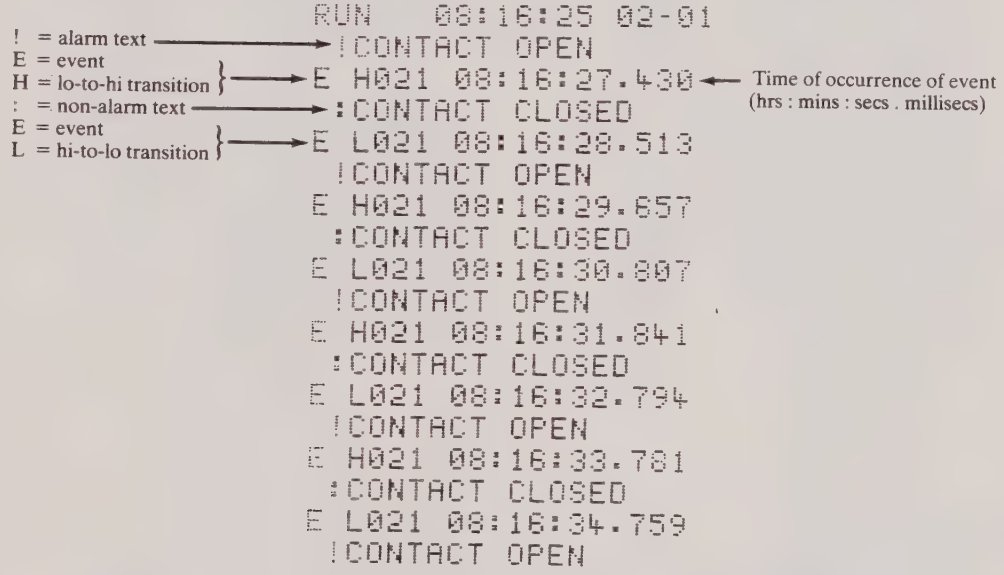
Event Indication

The example below shows the results obtained from an event channel (21). The event task is set to detect either event transition: i.e. an event = lo-to-hi or hi-to-lo transition.

The task is being used to detect the movement of a switch contact, and two texts have been applied to the event channel in order to indicate contact open/contact closed. The texts have been defined as follows:

Non alarm text = CONTACT CLOSED
Alarm text = CONTACT OPEN

A non-alarm text is output with any hi-to-lo transition event; an alarm text is output with any lo-to-hi transition event (although text is referred to as non-alarm and alarm, all events are regarded by the logger as alarms).



Listing

Using the LIST facility, a listing of any Channel, Task, Conversion, Text Definitions or Program can be produced.

Channel Definition
(for Channel 1)

```
--------------------
CHANNEL  001
measurement Volts
 DC
conversion 1
alarm Limits
high  0.0250000
low  1.000E-3
o/p 000,000
analogue o/p  0
text 1
alarm text  6
```

Conversion Definition
(for Conversion 1)

```
--------------------
CONVERSION  001
type Polyn
a,b,c,d  5.00000, 4.
00000, 3.00000, 2.00
000
units PSI
```

Text Definition
(for Text 1 and 6)

```
--------------------
TEXT  001
text SENSOR 1

--------------------
TEXT  006
text SENSOR 1 ALARM
```

Task Definition
(for Task 1)

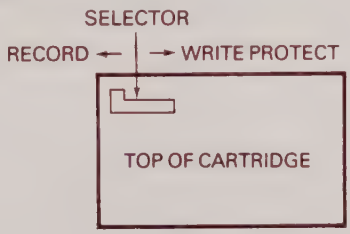
```
--------------------
TASK  001
title SCAN CHANS 1-3

Function Scan
trigger Timer
delay 00-00:00:00.0
scans 5
scan Inter
interval 00-00:00:05
.0
channels 001-005
rate 500/s
Processing On
data to be logged Al
l
header Always
output device Print
Format Full
tasks triggered 2,3,
4
tasks aborted 6
```

Program Listing
(For Task 1)

```
1000 MEAN(20)"MEAN"(
1)"VDC"
1001 DEV(20)"SD"(1)
1002 N=N+1
1003 FOR I=1 TO 20
1004 MEAN(I)=MEAN(I)
+C(I)
1005 DEV(I)=DEV(I)+C
(I)**2
1006 NEXT
1007 IF N=10 THEN
1008 FOR I=1 TO 20
1009 DEV(I)=MEAN(I)*
MEAN(I)/N-DEV(I)
1010 DEV(I)=SQR(ABS(
DEV(I)/N))
1011 MEAN(I)=MEAN(I)
/N
1012 PRINT MEAN(I),D
EV(I)
1013 MEAN(I)=0
1014 DEV(I)=0
1015 NEXT
1016 N=0
1017 ENDIF
```

3 CARTRIDGE RECORDER

The Cartridge Recorder accepts DC100 cartridges which may be used for read and write functions. Incorporated on the DC100 cartridge is a selector switch which can be set to Record or Write Protect:



When set to RECORD: Write and Read functions can be used.

When set to WRITE PROTECT: Read functions only can be used.

Write functions

Program MARK: reserves spaces for programs.

SAVE program: program in memory is recorded (this includes any initialised parameters such as unstrained gauge measurements).

LOG data: measurements are recorded

ERASE data: measurements are erased.

Read functions

DIRECTORY: a list of program numbers and names is output.

LOAD programs: a recorded program is output to memory.

COPY logged data: specified recorded data is output to an external device.

The functions required are selected using TAPE FACILITIES, see Chapter 9. Instructions for changing a cartridge are given on page 2.4.

Up to 500 kbyte of information can be recorded on one cartridge, in the form of "records". Each record occupies an identical length, irrespective of its content, but the length varies according to the format chosen. 50,000 records can be stored in Coded format, 25,000 in Compact and 12,500 in Full format.

The following pieces of information occupy one record each, in Full and Compact format:

A RUN or HALT marker.
A START or DONE TASK marker.
A date marker (every midnight).
A POWER FAIL or POWER BACK marker.
A result which has been stored in a pre-defined variable.
A result which has been stored in a user-created variable.

In Coded format, the first five also occupy only one record each, but results which have been stored in user-created variables occupy *two* records, as explained in Chapter 12, Section 14 "Storing and Replaying Processed Data and Programs." Read and write occur at 30 inches per second, and search for recorded information occurs at 60 inches per second. All recorded information is formatted to the same standard as ECMA 46 for DC300 cartridges. The cartridges are two-track, and both sides may be used for recording of logged data and programs. Tape length is 150 ft. Character blocks are 1024 bytes in length, and CRC is applied.

Adding Slave Expansion Units

When Slave Expansion Units are fitted to the logger, some previously unassigned memory is automatically reserved for them. This may mean that there is no longer enough free memory to accommodate a very long program, which has been recorded on cartridge prior to the Slave Units being fitted. Attempting to load such a program will produce one of the following results:

a. The program will load correctly, with the exception of any processing information, which will be ignored.

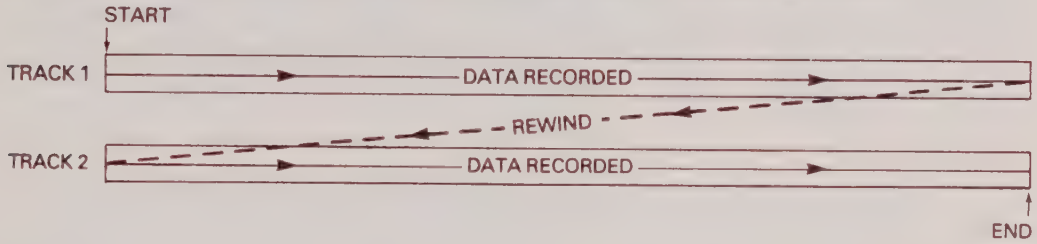
With the maximum number of 600 Channels fitted, the total program must not exceed 99 lines. This includes Channel definition, conversions, text, Task definition and processing information. To calculate memory availability with a lesser number of Channels, see the section 'Memory Capacity' in Chapter 12.

To save having to re-write the entire program, disconnect the Slave Expansion Units. The whole program should now load correctly, and hence can be edited in order to reduce its length to fit the smaller memory.

b. The whole program loads correctly, but when RUN is pressed, Error Code 152 is displayed. This indicates that there is insufficient memory to accommodate the processing program variables. As the whole program is now in the memory, it can be edited to reduce its length without disconnecting the Slave Units.

Recording Mode. The Recording Mode for logged data can be set to Normal or Cyclic.

Normal Data is recorded on track 1 until it is full, then tape rewinds to start of track 2. Data is recorded on track 2 until it is full (end of tape) and then recording stops. An 'end of tape' message automatically appears on the display and an audible warning will be heard.



Cyclic Data is recorded on track 1 and track 2 as in Normal mode except that, when the end of track 2 is reached, the tape rewinds and the data already recorded is overwritten by any subsequent data. In this way, previously recorded data on tracks 1 and 2 is continuously overwritten by newly logged data, leaving only the latest data on the tape. No 'end of tape' message is given.

If any logged data is output to the cartridge while it is rewinding, the data is temporarily held in a buffer which can store over 1,000 measurements. (Rewind time is approximately 30 seconds). When tape rewinding is completed, the measurements held in the buffer are automatically output to the cartridge.

Recording Format Measurements are recorded in 8-bit ASCII in one of 3 formats: Coded, Compact or Full (see Output Formats, page 11.11).

Coded Format The most economical use of tape space is achieved using Coded format; up to 50 000 records can be accommodated on one cartridge, as explained on the previous page. Coded format information consists of specially encoded printable ASCII characters. When the data is replayed the information is decoded, and reconstituted by the logger. The program used during recording must be reloaded to replay Coded data, in which case all the information normally present in Full format (text messages, etc) can be output.

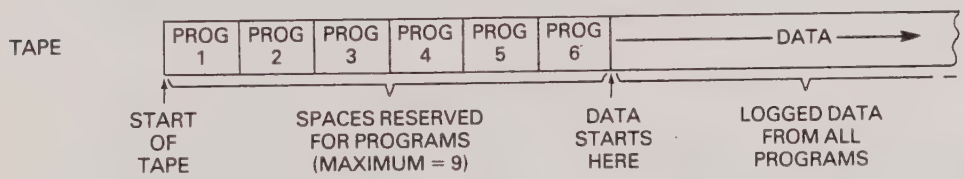
Cartridges recorded in Coded format would therefore normally be replayed in ORION, but can be replayed in other read stations if suitable decoding is applied.

Compact and Full Format Cartridges recorded in Compact or Full format can be replayed in any suitable read station without the need for any decoding. One cartridge can accommodate 25 000 records in Compact Format, or 12 500 records in Full Format.

Cartridges to be replayed in other read stations should ideally contain no programs (i.e. MARK = 0), this obviates the need to skip over program blocks when replaying the tape.

Programs A program consists of all setting-up information (complete data base), which is entered using the front panel controls or a remote terminal. Each program can contain up to 8 tasks.

The memory in ORION can contain one program, but up to 7 programs can be recorded on one cartridge. Because of the way in which programs are recorded – all the programs first, then all the logged data – it is essential to allocate spaces on the tape for the programs that are to be recorded. This is done using the MARK facility (part of TAPE FACILITIES) e.g. for 6 Programs, MARK = 6.



As each program is SAVED (recorded from memory onto the cartridge) it is recorded in the correspondingly-numbered reserved space. If, for instance, program 6 is SAVED first, it is recorded in space 6 leaving spaces 1-5 ready for programs 1-5. Each program consists of 30 kbyte, and occupies approximately 6% of the total recording space on the tape.

Program loading after Power Failure In the event of the logger memory being corrupted during a power failure (i.e. the internal back-up battery is exhausted) Program 1 can be automatically loaded into memory when power is restored. This provides a safeguard against total loss of a program, and allows the logger to re-start logging routines immediately power is restored.

This automatic loading of Program 1 (Program Booting) is conditional on the setting of SW2/2 on Controller, see page 3.6. If this facility is used, it is important to save the appropriate program as 'Program 1', as it is only this program which can be loaded automatically. The Program will also be loaded automatically at initial switch on of the logger.

The cartridge may be changed only if:
the logger is in the HALT mode (the tape may be in any position),
or
the logger is in the RUN mode, and the logger itself has initiated the PAUSE state as a result of the tape being completely filled. See page 2.4 for further details.

Cartridge Reader

An optional Cartridge Reader is available:

Qantex Series 1000 Cartridge Reader, Model 1000-1, sales code number 3416A.

This Cartridge Reader permits any cartridge recorded in ORION, in Full or Compact format, to be replayed.

4 OUTPUT FORMATS

4.1 ADDING TEXT

Any output result can, if required, be accompanied by a user-defined text (i.e. a short message). There are two types of text: non-alarm text and alarm text. One non-alarm text and /or one alarm text can be assigned to any input channel, or group of channels. Each text will then be automatically output with a result as follows:

Non-alarm text – output with any non-alarm result
Alarm text – output with any alarm result.

The texts are written using the Text Definition facility. Any mixture of non-alarm and alarm-only texts can be entered, up to a total of 40. Each text can consist of a maximum of 18 characters.

Each text, which has an exclusive number in the range 1-40, is attached to a specific input channel (or group of channels) by entering the appropriate text number in the Text part of the Channel Definition. A simple example is shown below.

Text Definition:	Text No. 1 = MESSAGE A Text No. 2 = MESSAGE B
Channel Definition for Channel n:	Non-alarm text number = 1 Alarm text number = 2

MESSAGE A and MESSAGE B are written and entered using the Text Definition, and the text numbers of these messages are entered in the Channel Definition for channel n. This will result in the following automatic action:

all non-alarm results from channel n will be accompanied by MESSAGE A
all alarm-only results from channel n will be accompanied by MESSAGE B

Channel n could, for instance, be a thermocouple channel used to measure temperatures on a boiler. MESSAGE A could state 'BOILER O.K.', whereas MESSAGE B, output when the temperature had exceeded an alarm level, could state 'BOILER OVERHEATING'.

Note that any defined text can be used as a non-alarm or alarm-only text, and any text can be used in any number of Channel Definitions.

The text that accompanies a processed result is dependent upon the user program, see Chapter 12. Results from Event inputs will be accompanied by text as follows:

Event, hi-to-lo transition – non-alarm text
Event, lo-to-hi transition – alarm-only text

All conditions for text output are summarised in the table below.

TEXT TYPE	RESULT TYPE			
	NON-ALARM LOG	ALARM LOG	EVENT hi to lo	EVENT lo to hi
Non-alarm	●		●	
Alarm only		●		●

● = text output

Note that text is only output in FULL Format, no text is output in COMPACT or CODED Format (Format is selected in the Task Definition).

4.2 GENERAL DETAILS

Messages and measured results are output by the logger to any one of the output devices selected in Task Definition (the devices include display, printer, cartridge, and devices connected to Interfaces). In addition to selecting the output device, the user may also select the output format; this can be Full, Compact or Coded Format.

Full Format This contains all information including text messages and results. It is intended for use when full information is required, at the expense of using more recording space (on print roll, etc). Full Format consists of 40-character strings.

Compact Format This contains full numeric information (results etc) but excludes text. Compact Format, which consists of 20-character strings, conserves recording space.

Coded Format This is intended primarily for recording data on the cartridge, but data recorded on the cartridge in Coded format may be saved to any other device. Coded format information contains results in binary form which are encoded into printable ASCII characters, in 10-character strings. Texts are not output in Coded format. Coded format takes up minimum tape space, and thus permits the maximum number of results to be recorded. Cartridges recorded in Coded format would normally be replayed in the logger only.

Monitoring Full or Compact format may be selected for monitoring, using the Monitor facility (MON). Note that the format selected for monitoring operates independently of the format for logged data, selected using a Task Definition.

Monitored results are distinguished from normal logged results as follows:

Monitored results are preceded by M
e.g. M 002 1.6554 V

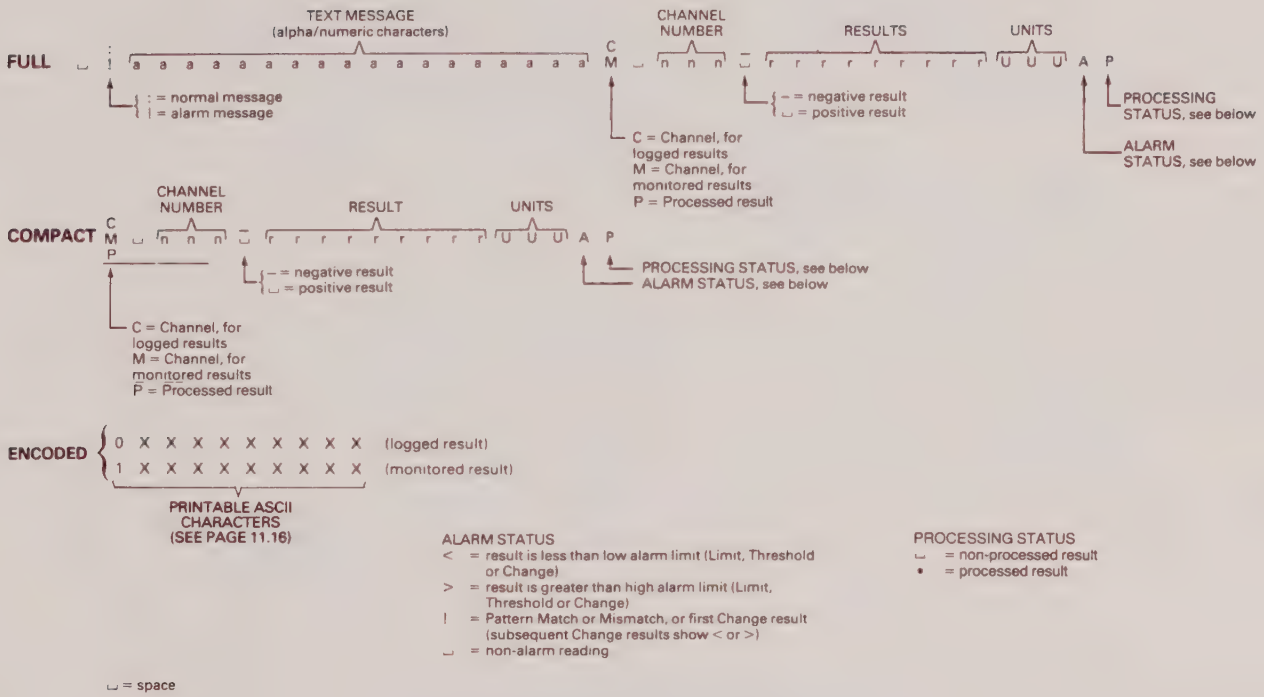
Logged results are preceeded by C
e.g. C 002 1.6554 V

Data output by the logger will be in Full, Compact or Coded format, as selected in Task Definition or Monitor. The content of messages in each of the three formats is standard, regardless of the output device used. 6 types of message/data are output by the logger; the type depends on logger action at any given time. The 6 types, which are output in any selected format, are listed below and detailed on the following pages.

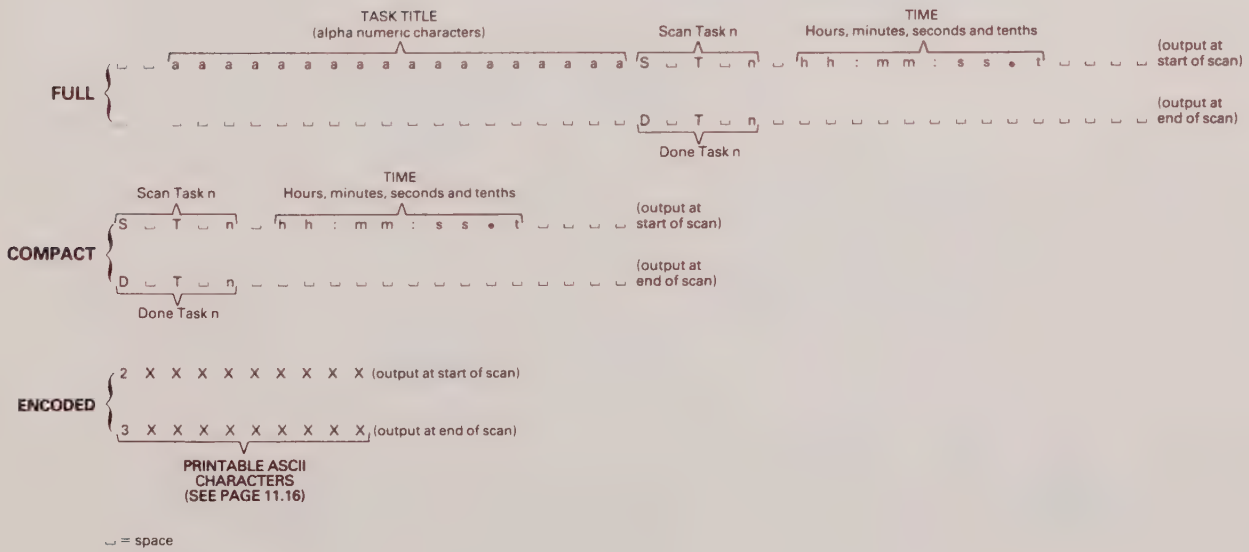
1. Numerical Result
2. Start and End of Scan Markers
3. Event Result
4. Run and Halt Markers
5. Power Fail and Power Back Messages
6. Date

Full detailed information on Coded Format is shown separately on pages 11.16 to 11.23.

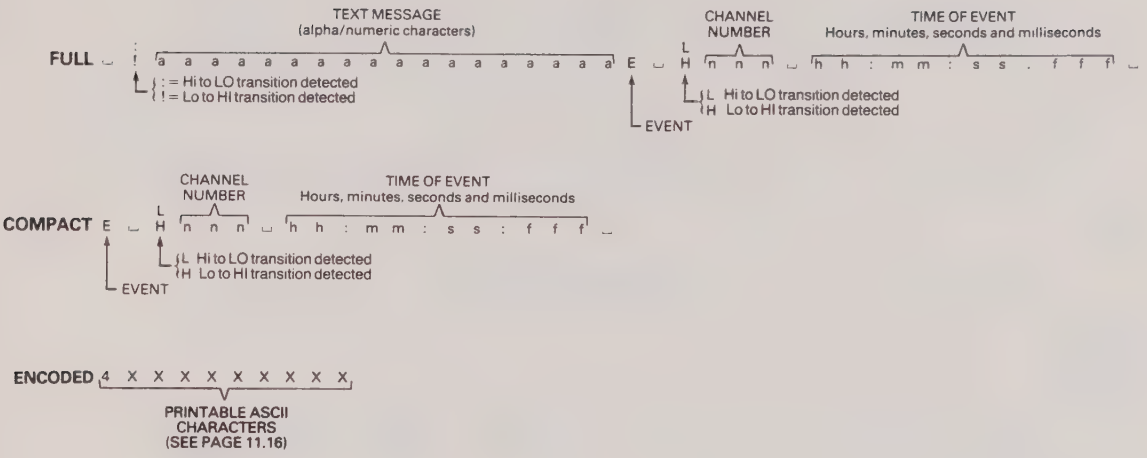
1. NUMERICAL RESULT


FULL
TEXT MESSAGE (alpha/numeric characters)
CHANNEL NUMBER
RESULTS
UNITS
: = normal message
! = alarm message
– = negative result
␣ = positive result
C = Channel, for logged results
M = Channel, for monitored results
P = Processed result
PROCESSING STATUS, see below
ALARM STATUS, see below
COMPACT
CHANNEL NUMBER
RESULT
UNITS
– = negative result
␣ = positive result
C = Channel, for logged results
M = Channel, for monitored results
P = Processed result
PROCESSING STATUS, see below
ALARM STATUS, see below
ENCODED
0 X X X X X X X X X (logged result)
1 X X X X X X X X X (monitored result)
PRINTABLE ASCII CHARACTERS (SEE PAGE 11.16)
ALARM STATUS
< = result is less than low alarm limit (Limit, Threshold or Change)
> = result is greater than high alarm limit (Limit, Threshold or Change)
! = Pattern Match or Mismatch, or first Change result (subsequent Change results show < or >)
␣ = non-alarm reading
PROCESSING STATUS
␣ = non-processed result
• = processed result
␣ = space

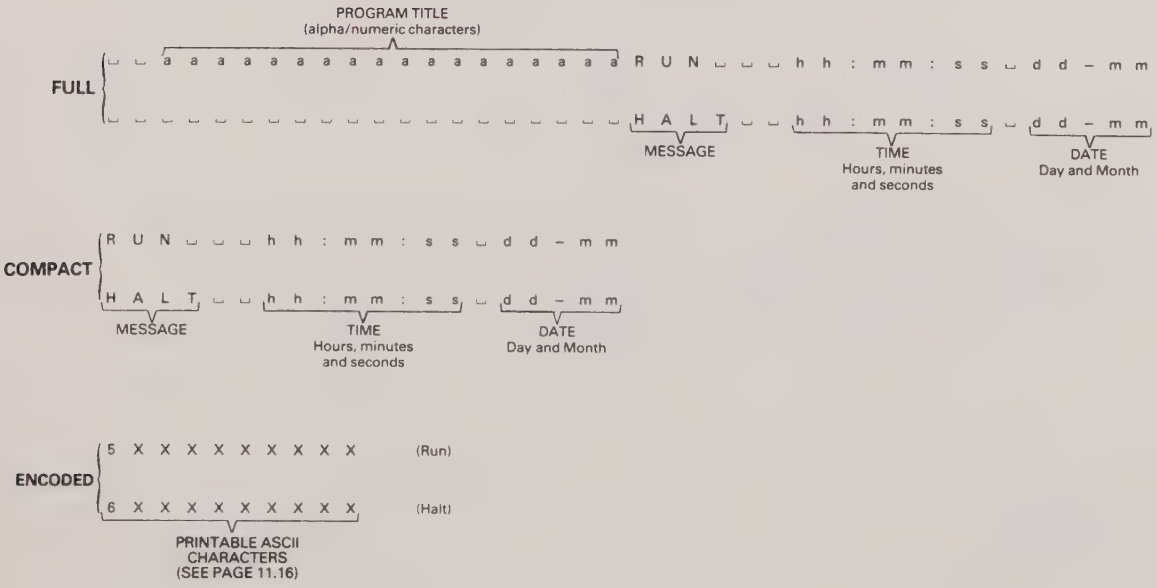
2. START AND END OF SCAN MARKERS


FULL
TASK TITLE (alpha numeric characters)
Scan Task n
TIME
Hours, minutes, seconds and tenths
(output at start of scan)
(output at end of scan)
Done Task n
COMPACT
Scan Task n
TIME
Hours, minutes, seconds and tenths
(output at start of scan)
(output at end of scan)
Done Task n
ENCODED
2 X X X X X X X X X (output at start of scan)
3 X X X X X X X X X (output at end of scan)
PRINTABLE ASCII CHARACTERS (SEE PAGE 11.16)
␣ = space

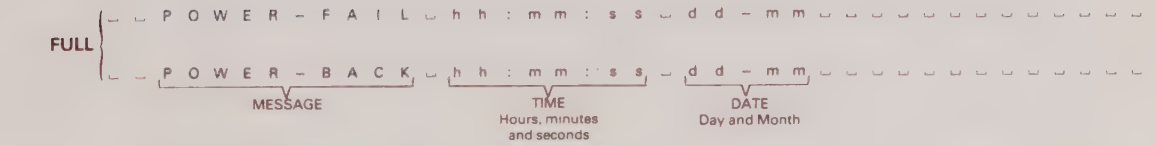
3. EVENT RESULT

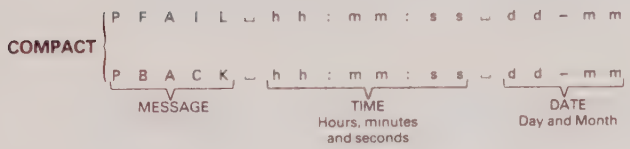


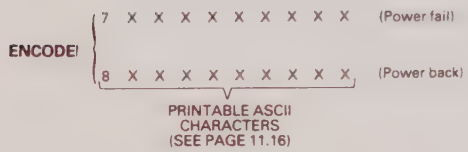
4. RUN AND HALT MARKERS



5. POWER FAIL AND POWER BACK MESSAGES

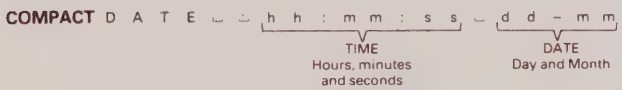


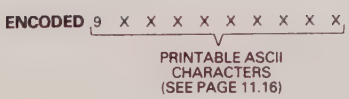




␣ = space

6. DATE





␣ = space

Specific Formatting

In addition to the general format information given on the preceeding pages, the following formatting details apply to the specified output devices only.

Cartridge Data is recorded in 1024 character blocks (the last 4 characters are always spaces). Partially filled blocks are filled with space characters.

Printer Terminal connected to PL1 on Controller (RS232) Data is output in lines of up to 80 characters, terminated by CR, or CR LF and 4 nulls as determined by SW2 on Controller. Measurements are packed 2, 4 or 8 per line depending on Format. (The VDU mode suppresses LF if line contains exactly 80 characters).

GP-IB devices, talker only mode Data is output in lines of up to 80 characters, terminated by CR LF. Measurements are packed 2, 4 or 8 per line depending on Format.

RS422 port b (on Serial Interface) Data is output in lines of up to 80 characters, termined by CR, or CR LF and 4 nulls. Measurements are packed 2, 4 or 8 per line depending on Format.

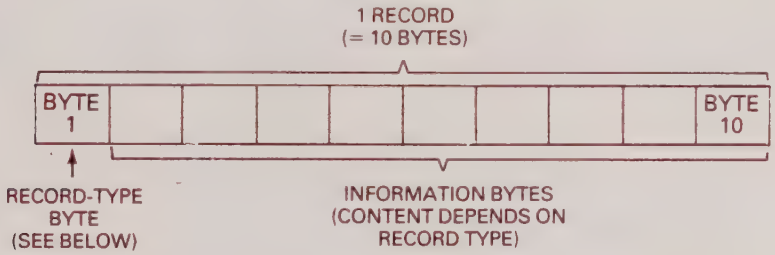
The terminating characters (e.g. CR, LF etc.) are additional to the data characters.

4.3 CODED FORMAT

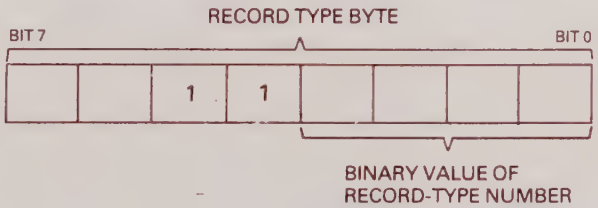
Records output in Coded Format consist of ten-byte packets. Each byte, which is encoded as a printing ASCII character, consists of: 1 parity bit (even parity) and
7 data bits

The most significant data bit (msb) is normally output as a '1'. This makes the byte into a printing ASCII character. When the binary value of the data alone exceeds 0111111, the msb assumes its true binary value.

All records consist of 10-bytes, although certain processed results consist of two records. The content varies according to the record type. The type, which depends on logger action at any given time, is indicated by the first byte in each record.



Record-Type Byte The four least-significant data bits indicate the type number.



RECORD TYPE NO.	
0	Numerical Result: normal logged result
1	Numerical Result: monitored result
2	Start of Scan Marker
3	End of Scan Marker
4	Event Result
5	Run Marker
6	Halt Marker
7	Power Back
8	Power Fail
9	Date

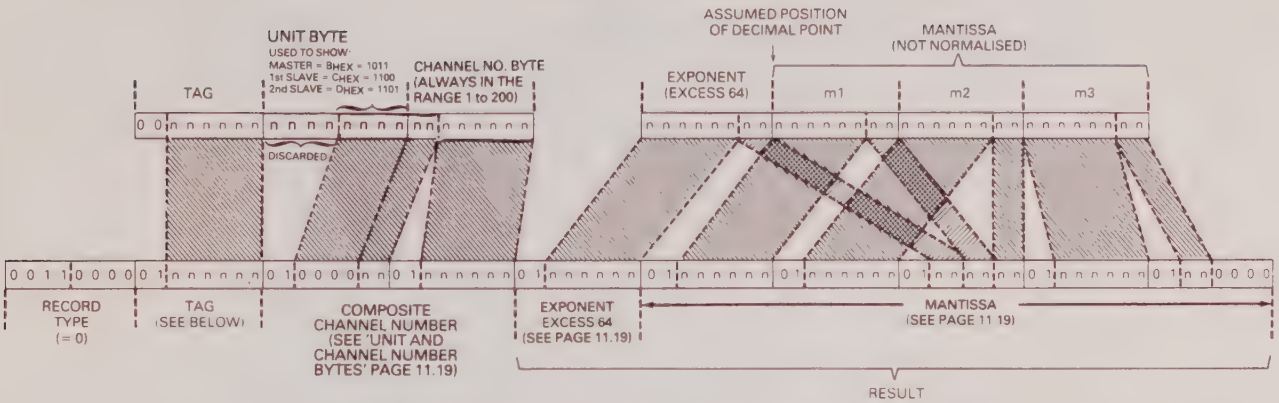
All Record types are detailed on the following pages. Each Record is shown in diagramatic form. The diagrams illustrate the overall content of the Records: the upper part of each diagram shows the numeric data which is generated within the logger, the lower part shows how the data is incorporated in the 10-byte Record which is output by the logger.

For general reference, a table showing all the ASCII characters and commands is included on page 11.24.

The various types of Coded format Records are listed below.

1. Numerical Result.
2. Start and End of Scan Markers
3. Event Result
4. Run and Halt Markers
5. Power Fail and Power Back
6. Date.

1. NUMERICAL RESULT

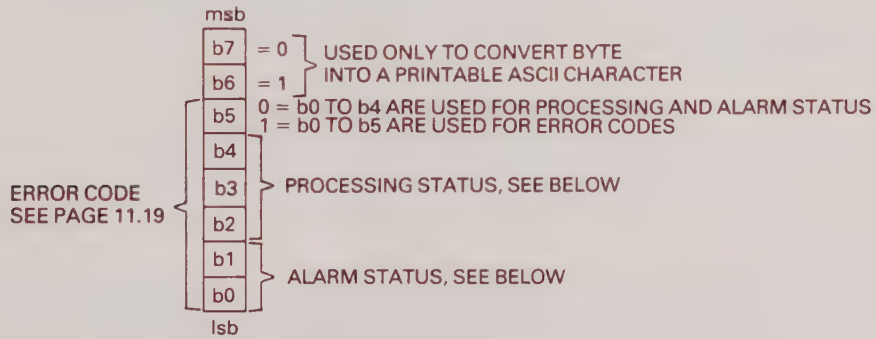


THE FORMAT FOR MONITORED RESULTS IS IDENTICAL, EXCEPT THAT THE RECORD TYPE (1ST BYTE) = 1

I.E. 0 0 1 1 | 0 0 0 1 — RECORD TYPE (= 1)

Tag Bytes These are included in Numerical and Event results. A tag byte can convey processing and alarm status, or error codes.

The tag byte is eight bits wide, although the two most significant bits are used only to make the byte into a printable ASCII character. The significance of each bit is described below.



PROCESSING STATUS

b4	b3	b2	
0	0	0	No processing
0	0	1	Processed result, where original Channel reading is overwritten
1	0	1	Derived processed result, where original Channel reading still exists

ALARM STATUS

b1	b0	*(a) For all measurements except Event detection*	*(b) For Event detection only*
0	0	Non-alarm result	Invalid condition
0	1	< Low limit	Logic Hi-to-Lo transition
1	0	> High limit	Logic Lo-to-Hi transition
1	1	Invalid measurement condition or Match/Mismatch detected	Invalid measurement condition

ERROR CODES

b5	b4	b3	b2	b1	b0	ERROR NUMBER	
1	1	1	1	1	1	1	Measurement overload
1	1	1	1	1	0	2	Measurement out of linearisation range
1	1	1	1	0	1	3	Invalid ambient temperature
1	1	1	1	0	0	4	Event total exceeds 65536
1	1	1	0	1	1	5	Timeout in external-module measurement (>20 secs)
1	1	1	0	1	0	6	Strain gauge channel not initialised
1	1	1	0	0	1	7	Input connector not fitted
1	1	1	0	0	0	8	Result not yet accumulated (Counter Timer module)
1	1	0	1	1	1	9	Counter Timer Channel is off

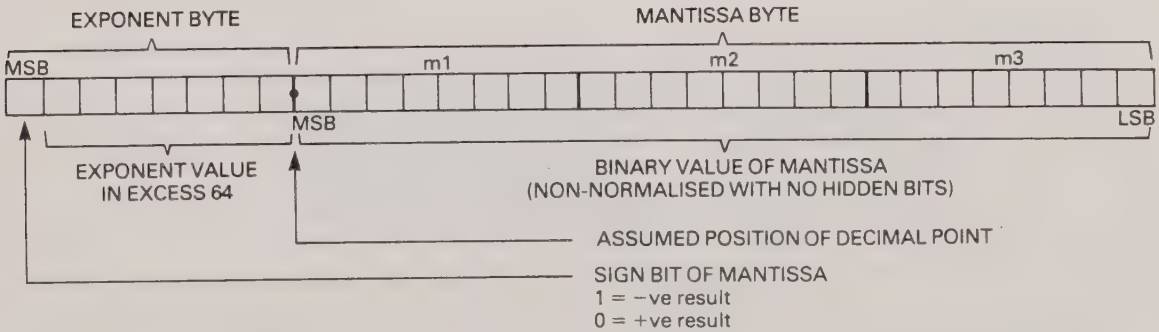
The error codes represent the 2's **complement** of the error number. If required, the error code may be extended by 2 bits to become an 8 bit 2's complement number, using external encoding. In this case bits 6 and 7 become '1', and the 8 bit number is read as 2's **complement** to obtain the error number.

Unit and Channel Number Bytes These produce a Composite Channel Number which indicates Unit (Master, 1st Slave, or 2nd Slave) and number within that unit (which is always in the range 1 to 200). However, in the case of the second record of a derived processed result, the Channel number is zero, indicating that the remainder of the record has no validity outside the logger.

Examples: Channel 005 (i.e. in the Master Unit) is shown as 1011 00000101
Channel 205 (i.e. in the 1st Slave Unit) is shown as 1100 00000101
Channel 405 (i.e. in the 2nd Slave Unit) is shown as 1101 00000101

Exponent and Mantissa

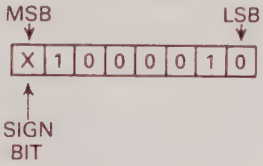
Each result is expressed in standard floating point format, as an exponent and mantissa (m1, m2 and m3). The sign bit for the mantissa is contained in the Exponent byte (msb).



Exponent Byte

The exponent byte contains the sign bit of the result (the msb) and the exponent value expressed as an Excess 64 number. To obtain the true exponent value, subtract 64.

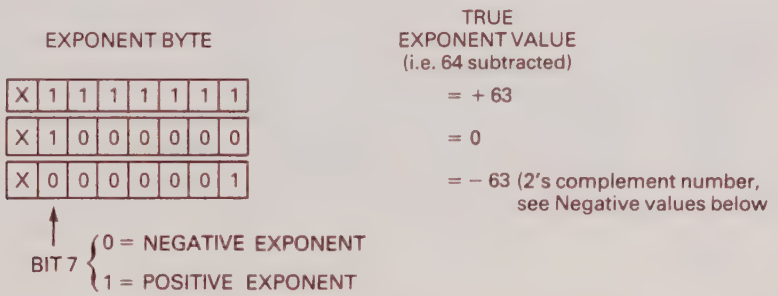
e.g. if the Exponent byte was



1. First take off the sign bit. The value then becomes: 1000010

2. Next, subtract 64 from the number; the value then becomes: 0000010
   Since $0000010_2 = 2$, the true value of the Exponent is 2.
   The exponent value (n) acts as a multiplier (2^n) on the mantissa.

The exponent value can be in the range +63 to −63. The bit patterns in the Exponent byte for true values of +63, 0 and −63 are shown below.


EXPONENT BYTE
TRUE EXPONENT VALUE (i.e. 64 subtracted)
X 1 1 1 1 1 1 1
= + 63
X 1 0 0 0 0 0 0
= 0
X 0 0 0 0 0 0 1
= − 63 (2's complement number, see Negative values below
BIT 7
0 = NEGATIVE EXPONENT
1 = POSITIVE EXPONENT

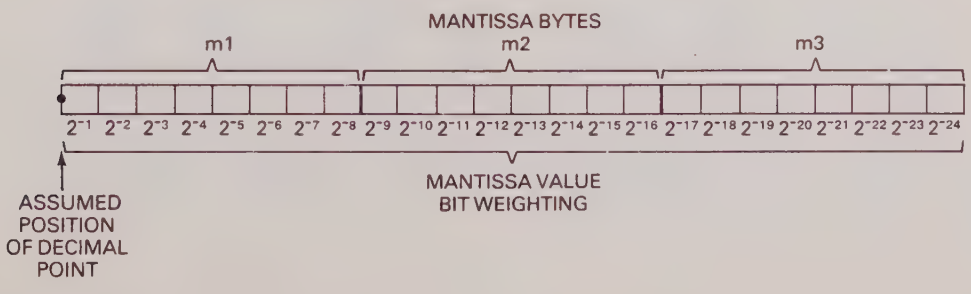
When the entire Exponent byte is zero, i.e. 0 0 0 0 0 0 0 0, the number (exponent + mantissa) is assumed to be zero.

Negative values A negative exponent is expressed in 2's **complement** form. To convert the bit pattern of a negative exponent (i.e. bit 7 = 0); first subtract 64, then apply 2's **complement**. This gives the positive equivalent of the negative number.

e.g. An exponent of −7 is given by the following bit pattern: X 0 1 1 1 0 0 1

Subtracting 64 from 0111001 gives 1111001. Applying 2's **complement** to 1111001 gives 0000111. This equals 7, indicating that the exponent value is −7.

Mantissa Bytes The mantissa is a floating point, non-normalised value with no hidden bits. The bytes m1, m2 and m3 should be considered as one long byte which contains the mantissa value. The bit weighting and assumed position of the decimal point is shown below.


MANTISSA BYTES
m1
m2
m3
2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6} 2^{-7} 2^{-8} 2^{-9} 2^{-10} 2^{-11} 2^{-12} 2^{-13} 2^{-14} 2^{-15} 2^{-16} 2^{-17} 2^{-18} 2^{-19} 2^{-20} 2^{-21} 2^{-22} 2^{-23} 2^{-24}
MANTISSA VALUE BIT WEIGHTING
ASSUMED POSITION OF DECIMAL POINT

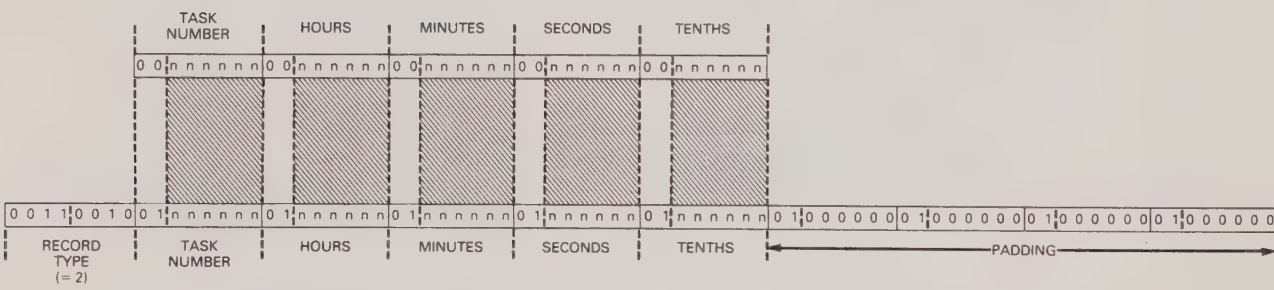
Mantissa polarity is given by the msb of the Exponent byte, 1 = −ve, 0 = +ve.

The effect of the exponent value (n) is to apply a multiplication factor of 2^n to the mantissa value. (This has the same effect as moving the decimal point and all bit weightings; i.e. a positive exponent n will move the decimal point n places to the right, a negative exponent n will move the decimal point n places to the left).

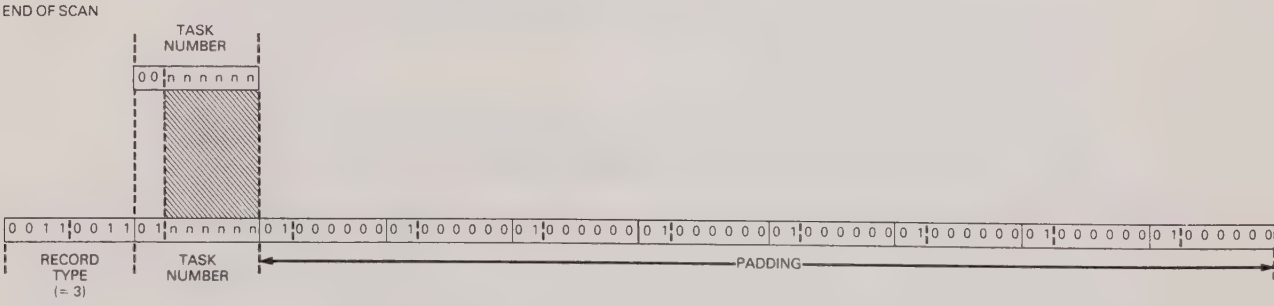
The Exponent byte and the Mantissa bytes are encoded into bytes 5 to 10 (inclusive) of the Numerical Result, as shown in the diagram on page 11.18.

2. START AND END OF SCAN MARKERS

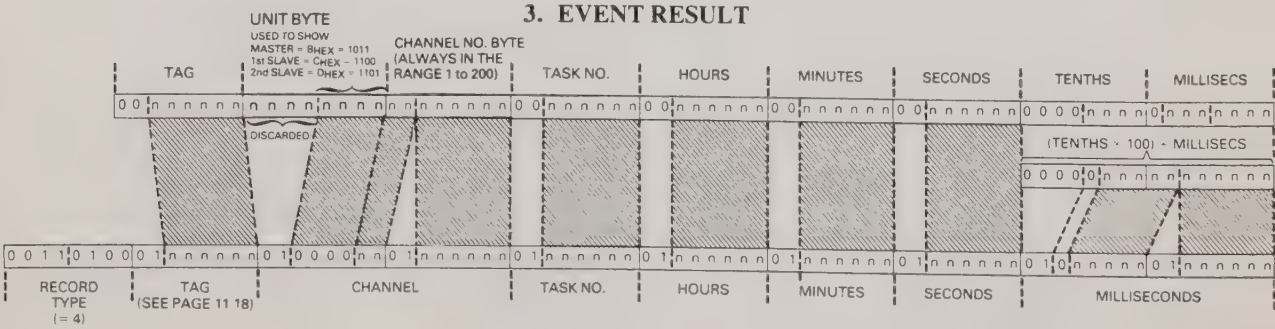
START OF SCAN



END OF SCAN

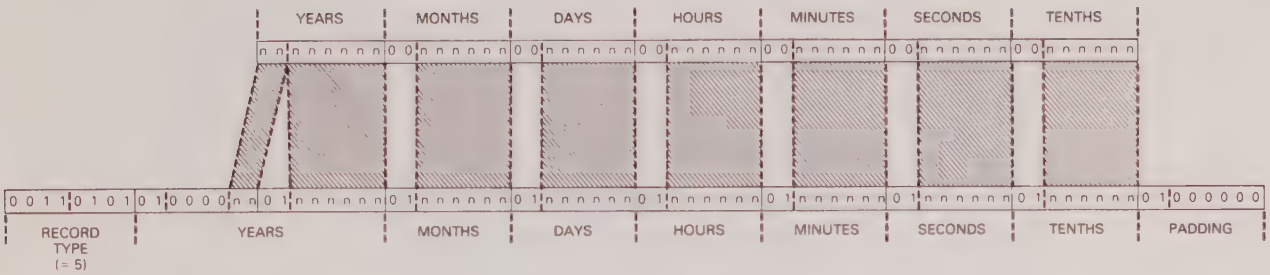


3. EVENT RESULT

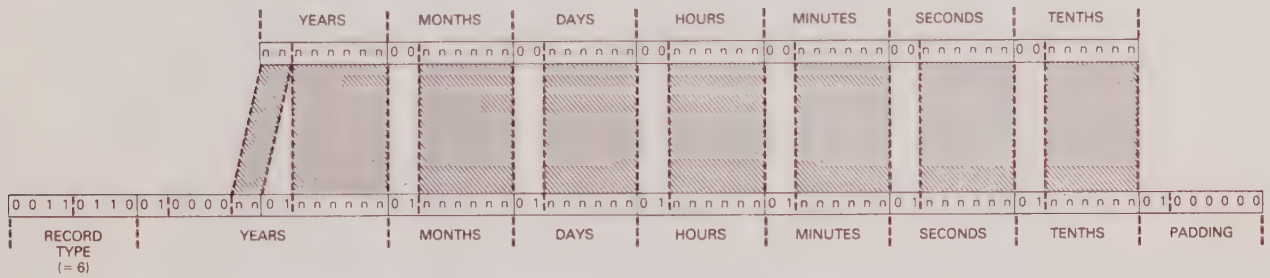


4. RUN AND HALT MARKERS

RUN MARKER

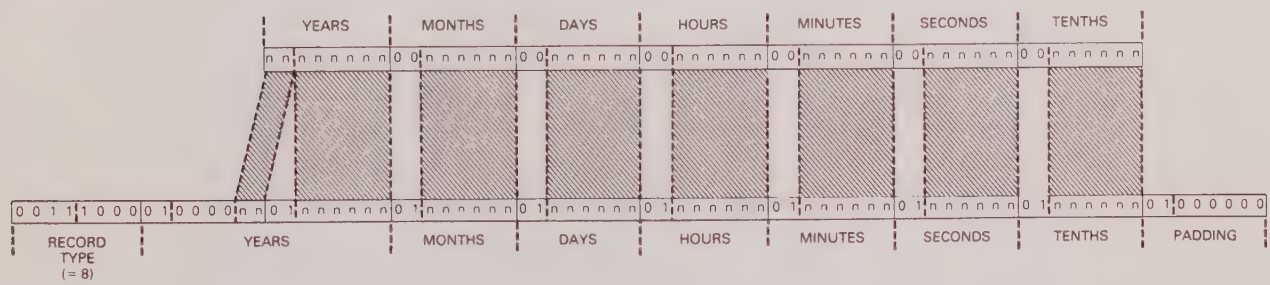


HALT MARKER

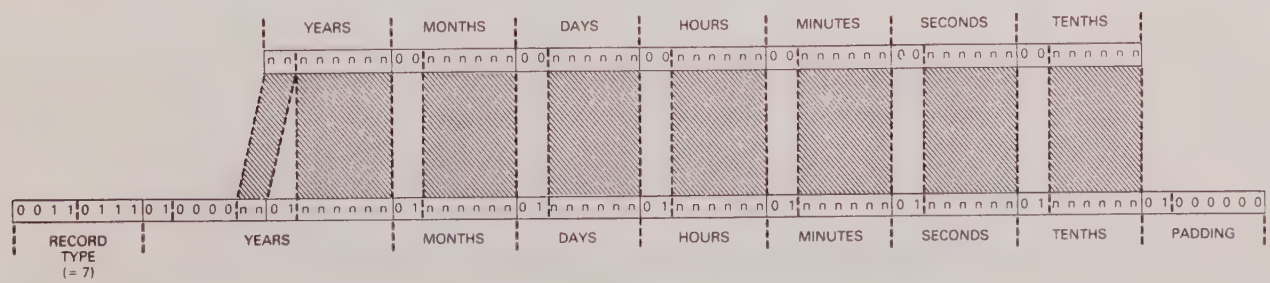


5. POWER FAIL AND POWER BACK

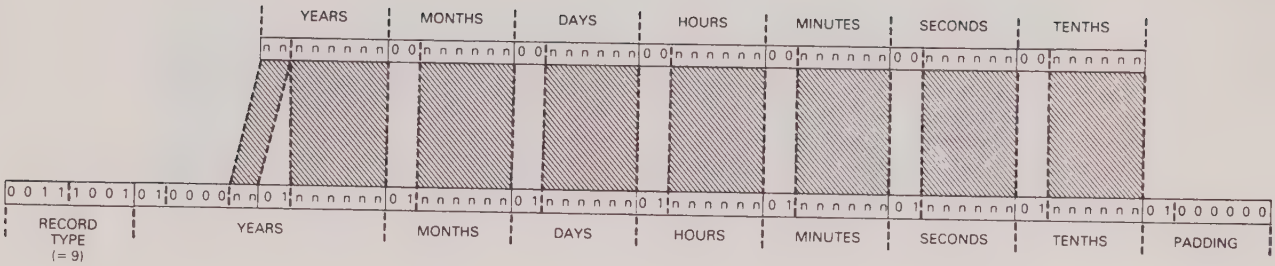
POWER FAIL



POWER BACK



6. DATE


YEARS
MONTHS
DAYS
HOURS
MINUTES
SECONDS
TENTHS
0 0 1 1 1 0 0 1
RECORD TYPE (= 9)
YEARS
MONTHS
DAYS
HOURS
MINUTES
SECONDS
TENTHS
PADDING

ASCII Codes

The table below shows the ASCII characters (and commands) produced by the 7-bit hexadecimal code used.

ASCII BYTE =	PARITY	b6	b5	b4	b3	b2	b1	b0

b6	0	0	0	0	1	1	1	1
b5	0	0	1	1	0	0	1	1
b4	0	1	0	1	0	1	0	1
b3 b2 b1 b0	0	1	2	3	4	5	6	7
0 0 0 0 0	NUL	TC1 (DLE)	SP	0	@	P	`	p
0 0 0 1 1	TC1 (SOH)	DC1	!	1	A	Q	a	q
0 0 1 0 2	TC2 (STX)	DC2	"	2	B	R	b	r
0 0 1 1 3	TC3 (ETX)	DC3	#	3	C	S	c	s
0 1 0 0 4	TC4 (EOT)	DC4	$	4	D	T	d	t
0 1 0 1 5	TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u
0 1 1 0 6	TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v
0 1 1 1 7	BEL	TC10 (ETB)	'	7	G	W	g	w
1 0 0 0 8	FE0 (BS)	CAN	(	8	H	X	h	x
1 0 0 1 9	FE1 (HT)	EM	)	9	I	Y	i	y
1 0 1 0 A	FE2 (LF)	SUB	*	:	J	Z	j	z
1 0 1 1 B	FE3 (VT)	ESC	+	;	K	[	k	{
1 1 0 0 C	FE4 (FF)	IS4 (FS)	,	<	L	\	l	|
1 1 0 1 D	FE5 (CR)	IS3 (GS)	-	=	M	]	m	}
1 1 1 0 E	SO	IS2 (RS)	.	>	N	^	n	~
1 1 1 1 F	SI	IS1 (US)	/	?	O	_	o	DEL

Chapter 12
Processing

Contents

		Page No.
1	Processing Facilities	12.3
1.1	Introduction	12.3
1.2	Summary of Processing Facilities	12.3
2	Introduction to Processing Programs	12.4
2.1	Processing in the Task Sequence	12.6
2.2	Line Numbers	12.8
2.3	Definition of a Variable	12.9
2.4	Introduction to ORION Variables	12.9
2.5	Creating and Running a Simple Program	12.10
2.6	Obtaining a Listing of the Program	12.12
3	Entering and Editing Programs	12.13
3.1	Entering Programs	12.13
3.2	Single and Multi-statement Lines	12.13
3.3	Statement Separators/Delimiters	12.14
3.4	Expressions	12.14
3.5	Assignments	12.14
3.6	The Processing Keys	12.15
3.7	The Edit Keys	12.15
3.8	Resequencing	12.18
4	Syntax and Error Information	12.19
5	Constants	12.20
6	Types of Variable	12.21
6.1	Simple Variables	12.22
6.2	Indexed Variables	12.22
6.3	Pre-defined Variables	12.24
6.4	User-created Variables	12.26
6.5	Global Variables	12.27
6.6	Rules Governing Variable Names	12.27
6.7	Applications of Pre-defined and User-created Variables	12.28
7	Mathematical Operators and Functions	12.32
7.1	Mathematical Operators and Priorities	12.32
7.2	Trigonometric and Exponential Functions	12.33
7.3	Relational and Boolean Operators	12.34
8	Control Structures	12.34
8.1	IF . . . THEN Statements	12.34
8.2	FOR . . . NEXT Loops	12.36
8.3	GOTO	12.38
9	Output Statements	12.39
9.1	Introduction	12.39
9.2	Output Formatting	12.39
9.3	Print Statements	12.40
9.4	Print Statement Extensions	12.45
9.5	Output Format Declaration	12.48
9.6	Result Numbering	12.50
9.7	Comparison of Print Statement Extensions and Output Format Declaration	12.51

9.8	Write Statements	12.54
9.9	KILL	12.57
9.10	SET . . . TO	12.58
10	Digital Inputs	12.61
11	Control of Other Tasks	12.62
11.1	Triggering Other Tasks	12.62
11.2	Aborting Other Tasks	12.62
12	Counters and Timers	12.63
12.1	Counters	12.63
12.2	Timers	12.64
13	Random Channel Access	12.64
14	Storing and Replaying Processed Data and Programs	12.68
15	Memory Capacity	12.69
16	Vocabulary of Functions	12.69
17	Example Programs	12.70
17.1	Contents	12.70
17.2	Introduction	12.71
17.3	Single Channel Average over 5 Readings	12.72
17.4	Average of 3 Channels on Every Scan, with a Cumulative Average of Each Channel Every 5 Scans	12.73
17.5	Maximum and Minimum Readings, on One Channel over 10 Scans	12.74
17.6	Output on Change, for One Channel	12.75
17.7	Output on Change, Giving Old and New Values and the Difference Between Them	12.76
17.8	Warning if Channel Reading is Outside Limits	12.77
17.9	Alarm Threshold	12.78
17.10	Walking Window Averages	12.79
17.11	Walking Window Average on One Channel	12.80
17.12	Walking Window Average on 3 Channels, over 5 Readings	12.81
17.13	Mean Reading and Standard Deviation of 20 Channels, Over 10 Scans	12.82

1 PROCESSING FACILITIES

1.1 INTRODUCTION

ORION *delta* Channel results obtained in a Scan Task, may be manipulated by the internal processor under the control of a user program. This program is included as part of the Task and uses BASIC-like statements. A separate program may be appended to each of the 8 Tasks.

The program treats the results as variables, offering unlimited possibilities for obtaining derived results from a number of measured channels. Further possibilities are data reduction programs, flexible cross-triggering of Tasks and sophisticated output control using both analogue and digital signals.

1.2 SUMMARY OF PROCESSING FACILITIES

The Processing Facilities apply only to the specific Channel(s) when they are used in the relevant Task.

The following facilities are available:

a. Individual channel results can be modified by various mathematical operations, such as scaling, conversions involving logarithms and trigonometric functions etc. These operations can augment, or replace, the conversion facilities available under "Channel Definition".

   For example, the output of a transducer connected to Channel 1 may have been converted to engineering units as part of the Channel Definition sequence, giving a value x. The Processing Facility then allows this reading to be converted to, say, x^2, $\log x$, $\sin x$ etc.

b. Cross-channel calculations, such as ratios, power measurements ($V \times I$) etc.

c. Calculations involving time, such as averages, statistical behaviour, control algorithms etc.

d. "PRINT" statements to direct selected results to Display, Printer and peripheral interfaces.

e. Making recorded results more meaningful by adding explanatory text to them.

f. The cross-triggering of other tasks under special circumstances.

g. The cross-aborting of other tasks under special circumstances.

h. The control of analogue and digital outputs.

i. Decision Processing. The logger can test results such that, IF a defined condition is found, THEN a prescribed action is taken, ELSE an alternative action is implemented. This decision making may include combinations of conditions that involve AND and OR. It can be applied to analogue and digital results.

2 INTRODUCTION TO PROCESSING PROGRAMS

This Section introduces the complete beginner to certain essential fundamental concepts and also demonstrates how to write, and run, a simple program. More advanced programming techniques are explained in later Sections of the Chapter.

A processing program consists of a series of lines containing one or more statements, each statement using common English words and familiar mathematical notation. Any of the facilities listed in Section 1.2 can be implemented by the processor, under the control of an appropriate program.

The original measurements may have already had some form of processing applied to them automatically. For instance, all thermocouple and platinum resistance thermometer readings are stored as temperatures in degrees Centigrade, rather than as a number of millivolts. Similarly, strain gauge readings are converted to, and stored as, units of μ strain.

In addition, the Channel Definition menu, as shown in Chapter 9, offers up to 40 optional Conversions, but these are relatively limited in their application compared to the much greater range and flexibility afforded by the processing programs. Note particularly that when a Conversion is selected for a given Channel, it is applied to the results from that Channel in *every* Task which scans it. Each processing program, however, acts *exclusively* on the results from the Channels scanned in the Task which contains that program.

There are two fundamental rules to remember:

1. *Each Channel result is stored in a completely independent location. Therefore, if all 8 Tasks scan the same Channel, there will be 8 independent results stored, each one accessible only to the Task which created it.*

   For example, if Channel 1 is scanned in Tasks 3, 4 and 5, there will be three results stored, viz:

   1. Result from Channel 1 when scanned in Task 3.

   2. Result from Channel 1 when scanned in Task 4.

   3. Result from Channel 1 when scanned in Task 5.

2. *No Channel result can be processed until that Channel has been declared in the relevant Task.*

   This declaration requires that the following three conditions are met:

   a. An Input/Output Module must be present in the appropriate slot. If this is a 20-Channel Module, such as the Reed Selector, then all 20 Channels are available for declaration. However, if it is a 5-Channel Module, such as the Counter Timer, then only the relevant 5 Channels are available.

   b. The Channel must be "configured" in the Channel Definition. This is achieved by pressing CHAN, then entering the desired Channel numbers followed by the selection of one of the Sensor menus. The rest of the Channel Definition is optional but this initial part is essential, as the default state is "unconfigured" and hence the Task will not be able to scan the Channel.

c. The Channel must be included in the Task Definition, in reply to "Enter Channels to be scanned".

If the above declaration is not carried out, and an undeclared Channel is included in a processing program, Error 174 "Channel Not Present" will be produced when scanning is initiated by pressing RUN.

The lines of program are built up in the Display by using combinations of the PROCESSING keys and the keypad characters. Each line is transferred to the processor's memory by pressing the ENTER key. Errors can be corrected, and modifications incorporated, by using the eight EDIT keys. In addition to running the program, the processor also checks it for correct syntax (explained in Section 4) and warns the user of any errors. A list of Error Codes is given in Appendix 1.

The simple high-level language statements can be composed, edited and reviewed from the Front Panel. They can also be composed and reviewed remotely via the interface ports. Editing can be performed remotely by creating a complete new line bearing the same number as the one to be replaced. The programs can be saved on, and subsequently reloaded from the integral cartridge unit of ORION D.

A simplified block schematic of ORION *delta* is given in Fig. 12.1. This diagram illustrates how different areas of memory are allocated to specific functions. It also indicates how the processor can take the original Channel results, manipulate them, then re-store them in the same, or different, variables, under the direction of the processing program.

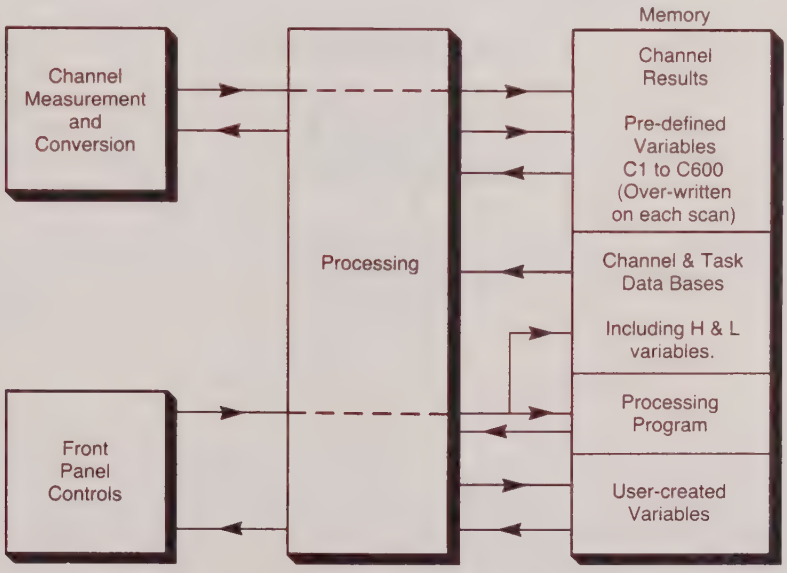


Fig 12.1 *Simplified Block Schematic of ORION delta.*

2.1 PROCESSING IN THE TASK SEQUENCE

The decision as to whether to process the measured data or not forms part of the Scan Task Definition, as shown in Fig. 12.2 below.

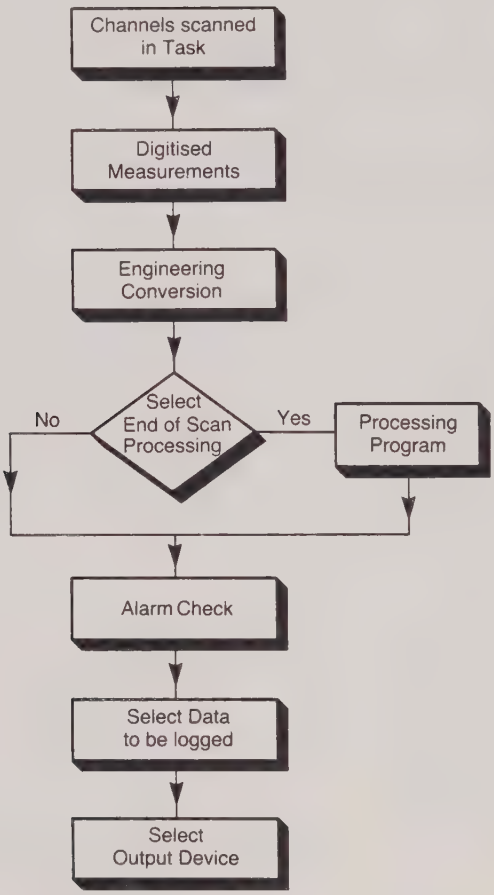


Fig 12.2 *Simplified Data Flow Chart.*

The Display will show: "Select end-of-scan processing", with two soft keys entitled "On" and "Off": If "On" is selected, processing will be performed on the Channel readings within that specific Task only. The processing of Channels in one Scan Task is completely independent of the processing of the same Channels in any other Task.

Note that only the *decision* whether to process data or not is included in the Task Definition. The processing program itself can be written into ORION's memory at any time, as described in Section 2.5 "Creating a Simple Program" and Section 3 "Entering and Editing Programs". If the Task Definition selected processing "Off", and a processing program is written at a later date, this program can be incorporated in the Task by simply halting the logger, recalling the Task Definition line "Select end-of-scan processing" to the Display, and pressing the soft key "On".

Similarly, processing can be deleted from the Task, by switching "Select end-of-scan processing" to "Off". The program is still retained in memory, and can be re-incorporated in the Task by selecting "On" again.

The relative timing of the scanning and the processing is shown in Fig. 12.3 below.

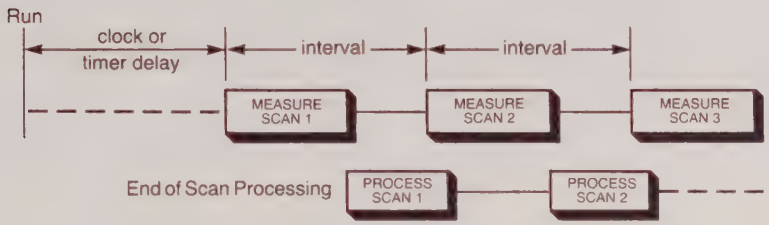


Fig 12.3 *Processing Timing Diagram.*

The above diagram shows that no processing will take place until all the Channels in that Task have been scanned.

Note also that when the logger has been halted, pressing RUN again has two main effects.

1. The processor *immediately* reads the programs and:

   a. Checks for syntax errors, as described in Section 4.

   b. Allocates room in the memory for any user-created variables etc. (Variables are introduced in Section 2.3).

2. After the processor has checked through the programs, which takes only a few microseconds, and any delay included as part of the Task Definition has elapsed, scanning commences.

2.2 LINE NUMBERS

The first item in each line of a program is a four-digit number, e.g. 1013, where the first digit indicates the number of the Task of which this program forms a part, in this case Task 1. The remaining three digits, 013, indicate that this is the 14th line of the program. The line numbers are used as labels or addresses within the program and when writing or editing programs, as explained in later sections.

Each program should start with line 000, and may extend up to line number 999, as illustrated in Figs. 12.4 and 12.5 below. However, the total number of lines that can be included in a processing program is determined by the memory capacity, specified in Section 15.

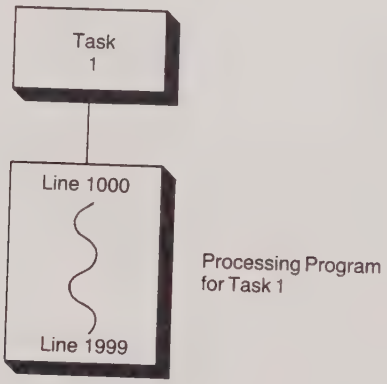


Fig 12.4 *Processing Program Line Numbers for Task 1.*

The remaining 7 programs would have their lines numbered as indicated in Fig. 12.5 below:

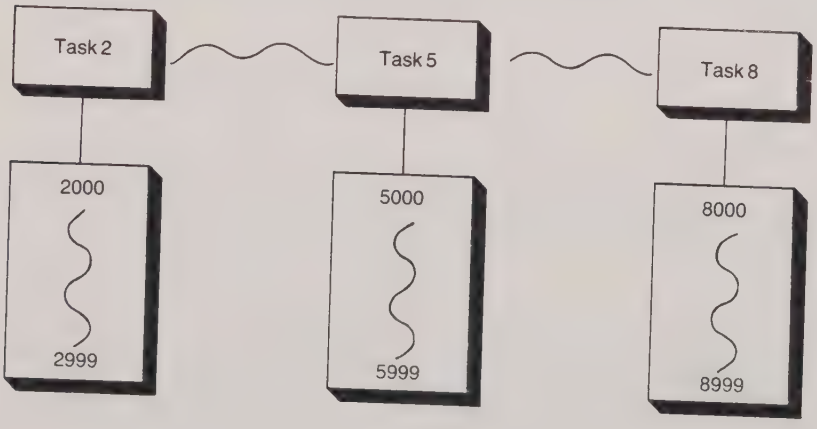


Fig 12.5 *Line Numbers for Other Tasks.*

Further information on the use of Line Numbers is given in Section 2.5 "Creating A Simple Program" and Section 3.1 "Entering Programs".

2.3 DEFINITION OF A VARIABLE

The term "variable" is used frequently throughout this chapter, and is defined here as:

"the name given to a location in the ORION memory in which a piece of information may be stored".

This information could be the result of a Channel measurement, a High or Low Channel limit, a processed result, an intermediate value, etc.

The name of the location also acts as its address. Hence, whenever the processor sees the name of particular variable in the program, it will either store information in, or retrieve information from, that specific memory location and no other. When processing is complete, a record may be obtained of the contents of the variables by outputting them to a selected device such as the printer.

2.4 INTRODUCTION TO ORION VARIABLES

Variables can be classified under several descriptive titles as explained in Section 6 "Types of Variable". However, for the moment, the user need consider only two types:

1. **Pre-defined Variables.** These variables are termed "pre-defined" because they have had their names permanently reserved for them, as a design feature of ORION. There are three groups of them, the most important being the variables C1 to C600. As shown in the block schematic of Fig. 12.1, these are the variables in which the unprocessed Channel results are automatically stored by the logger during a scan. These results are over-written on each new scan. The processor always goes to these variables to obtain the original Channel results, when the program instructs it to process them.

   The unprocessed Channel results are automatically stored in their correct locations, e.g. Channel 20 result is always stored in the pre-defined variable C20. Remember that, although, say, Tasks 3, 4 and 5 all include "Channel 1" in their scans, three completely independent results will be stored. Each of them will be stored as "C1", so there would appear to be *three* pre-defined variables all bearing the same name. However, each of these variables is accessible for processing only *within its own Task*, so there is, in fact, no confusion.

   During processing, the user may elect to store intermediate results back in the pre-defined variables, or to create new ones for them, according to whether the original results are still required or not. Other factors affecting this decision are discussed in Section 6.7.

   The user is reminded that Channel results cannot be processed as part of a Task unless they have already been declared in that Task, as explained in Rule 2 in the introduction to this Section.

   The other two groups of pre-defined variables consist of the High and Low Limits which can be applied to the Channel results. These are described in Section 6.3 "Pre-defined Variables", which also contains more detail on the variables C1 to C600.

2. **User-Created Variables.** Unlike the previous type, these variables have no permanent existence in the memory. They can be employed freely throughout the program, and are created by the user who assigns names to them according to the rules defined in Section 6.6. Immediately RUN is pressed, and before scanning commences, the processor reads through the program, and allocates sufficient space in the memory for all the user created variables that it finds. Thereafter, the variables can be used as required, for intermediate workspace and to store the processed results created after each scan.

User-created variables remain in existence only until RUN is pressed again to halt the logger. After this, they, and their contents, disappear from the memory. However, as long as the program information remains in the memory, the user created variables will have space re-created by the processor each time that RUN is pressed to start scanning.

2.5 CREATING AND RUNNING A SIMPLE PROGRAM

To gain familiarity with the processing program controls, the user should enter the following simple program, as explained in detail below. The object of the program is to take the result, which has been stored during the scan in the pre-defined variable C1, multiply it by 10, then store the new, processed result in the user-created variable A. The processed result is then printed.

The lines must be typed exactly as shown, for example do not press the SPACE key unless so instructed. The full syntax is described in Section 4.

```
1000  A=C1*10
1001  PRINT A
```

Note: this program assumes that Task 1 scans a dc voltage on Channel 1, as explained in Chapter 9 Section 3.2, Channel and Task Definitions.

Remember that any keying errors can be rectified by using the EDIT keys, as described in Section 3.7.

1. Press the DEFINITION key INIT.

2. Press the soft key EDIT.

3. The top line of the Display will now show:

   "Enter program line to being edit"

   Type the line number 1000 on the keypad, whereupon 1000 will appear in the bottom line of the Display.

4. Press ENTER. The line number 1000 will move to the top line, leaving the rest of the Display clear to receive the program information.

5. Type A=C1*10 on the keypad, remembering to use the SHIFT key before and after A and C. The characters will appear in the Display as they are typed, following the line number 1000.

   In this line of program, a new variable called A has been created by the user and the processor will automatically allocate a place in the memory for it. The line means: "The value stored in variable A becomes the value of Channel 1 multiplied by 10."

6. Press ENTER. The previously displayed line will be transferred into the processor memory and the next line number, i.e. 1001, will automatically appear in the otherwise empty Display.

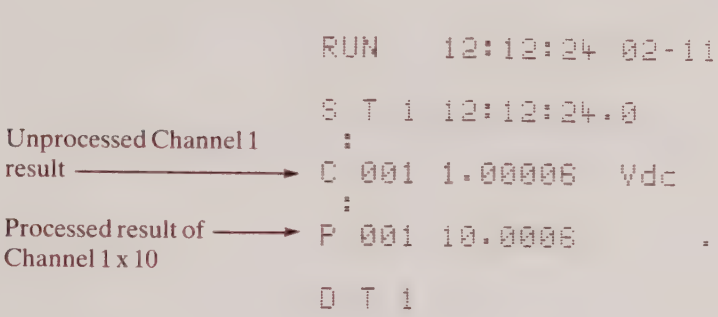
7. Press the PROCESSING key PRINT followed by the keypad key A. All the PROCESSING key words are automatically followed by a space; there is no need to add one. Press ENTER.

This line instructs ORION to print the value contained in variable A, and completes the simple program. Pressing ENTER will have caused line number 1002 to appear in the Display, but this is not required. Hence, to retrieve the "Ready for command" message:

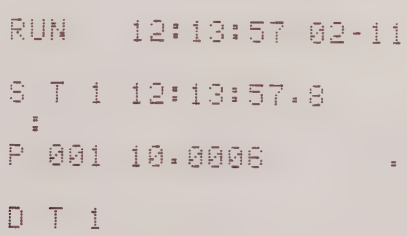
8. Press INIT.

9. Press EXIT.

Pressing the RUN key will now produce a print-out similar to that shown below:



In this example, the data to be output was determined by the Task Definition, where "Select data to be logged" was set to "All" and "Select output device" was set to "Print". However, data reduction could be introduced at this stage by setting "Select data to be logged" to "Proc", which will produce the following print-out, in which the unprocessed result has been suppressed.



The information accompanying a processed result can be considerably extended, as described in Section 9 under the headings "Print Statement Extensions" and "Output Format Declaration".

The user is reminded that *all* Channels referred to in the processing program *must* be declared in the Task. If an attempt is made to RUN the processing program shown above as part of Task 1, without first declaring Channel 1, the Display will show:

TASK 1 ABORTED – ERROR 174 ON LINE 1000

As shown in Appendix 1, Error 174 means "Channel not present", and the logger also helpfully identifies the line on which the error has been detected.

2.6 OBTAINING A LISTING OF THE PROGRAM

To obtain a printed record of the program, proceed as follows:

1. Press the DEFINITION key LIST.
2. Press the soft key Progr.
3. The Display will show:
   Enter program line numbers
4. Type 1000-1001 ENTER.
5. The Display will show:
   Select output device
6. Press the soft key Print
7. The printer will produce:

```
1000 A=C1*10
1001 PRINT A
```

Note that if, in Step 4, the ENTER key is pressed without first declaring any line numbers, *all* programs in the memory will be listed.

Further applications of the LIST facility are given in Chapter 11.

Programs can also be stored on a tape cartridge, as described in Section 14.

3 ENTERING AND EDITING PROGRAMS

3.1 ENTERING PROGRAMS

The method of entering a simple processing program is described in Section 2.5. More comprehensive programs are entered in a similar manner, a line at a time, by using combinations of:

a. the words on the PROCESSING keys

b. the keypad characters

c. the EDIT keys, for correcting keying errors.

The keypad has already been described in Chapter 9, and the user should by now be familiar with it as a result of setting up the Channel and Task Menus. The other keys are described in sections 3.6 and 3.7 and the associated syntax rules are given in Section 4.

The user is advised to write the program out fully on paper before entering it in the logger.

Consecutive line numbers should be used, as these are offered automatically by the processor, as described in Section 2.5 "Creating a Simple Program". Any omissions can be corrected later by means of the LINE INSERT key, as described in Section 3.7 "The Edit Keys".

Use of Line Number *x*999
The line number *x*999 is the highest that can be used in Task *x*. Some programmers prefer to allocate the highest line number available to the final line of program, even where this leaves many intermediate lines unused. However, this technique should be used with caution, as it precludes the use of the LINE INSERT key, as explained later in this section under "Resequencing".

3.2 SINGLE AND MULTI-STATEMENT LINES

Each line of program can consist of one or more statements, provided that the line length does not exceed the capacity of the Display. Although the Display can theoretically accommodate up to 71 characters of program in its two lines, the programmer should keep well below this number to allow for internal expansion of the line when coded for transfer to the memory. The expansion factor varies according to the line content.

An example of a multi-statement line is shown below. It contains two statements:

However, for relatively short programs, it is preferable to assign a new line to each statement, e.g., the above line would become:

```
4008 SCALE = 1.74

4009 OFFSET = 2.3
```

This technique improves the clarity of the program presentation, and also avoids the risk of exceeding the capacity of the Display.

However, multi-statement lines provide the most efficient utilisation of memory capacity. As stated in Section 15, each line of program, irrespective of its length, occupies a fixed amount of memory. For this reason, long programs may have to employ multi-statement lines.

3.3 STATEMENT SEPARATORS/DELIMITERS

Where multi-statement lines are used, colons must be inserted as separators between each pair of statements. The only exceptions to this rule are statements of the type IF . . . THEN, where the word THEN acts as an automatic statement separator. Also, no separator is needed after the final statement in a line, as it is automatically completed by pressing ENTER. However, for consistency, a separator can be inserted after every statement, if so desired; it does no harm.

An example of a multi-statement line, including a separator, is given in the previous section.

3.4 EXPRESSIONS

The term "expression" appears in later sections of this Chapter, where certain programming statements are given in their general form. An expression is defined here as:

"a combination of variable and function names and constants, which are related to each other via an operator, (as defined in Section 7)".

In its most elementary form, an expression may be simply a constant, e.g. 2, or a variable, e.g. TEMP. However, the more usual form includes an operator, such as TEMP*8, (i.e. TEMP multiplied by 8), C1>5, and (C1+C2)/2.

3.5 ASSIGNMENTS

In BASIC, from which the ORION *delta* software language is derived, the "=" sign has two distinct uses:

a. In "assignments", where it means:

"the expression to the left of the "=" sign is assigned the value of the expression to the right".

A simple example is A=8, where the the variable A is assigned the value of the constant 8.

b. In equations, where it means:

"the expression to the left of the "=" sign is being tested for equality with the expression on the right".

This meaning is usually encountered in IF . . . THEN statements, described in Section 8.1 A typical application would be:

IF C1=C2 THEN . . . proceed accordingly.

The processor deduces the appropriate meaning of the "=" sign from the context of the line in which it appears.

The programmer can write either A=C1*C2 or C1=C1*C2, both are equally valid when it is remembered that they are assignments and not equations. The distinction to be made here is that A is a *user-created* variable whereas C1 is a *pre-defined* variable. Both of these types are introduced in Section 2.4 "Introduction to ORION Variables", and their relative merits are described in Section 6.7 "Applications of Pre-defined and User-created Variables".

An example to illustrate the dual use of the "=" sign is given in Section 10 "Digital Inputs".

3.6 THE PROCESSING KEYS

The eight dual-purpose PROCESSING keys provide, in convenient, one-push form, the words most commonly required for programs. The other, less-common statements etc. that the logger can recognise are constructed character by character from the keypad.

The PROCESSING keys are used in conjunction with the SHIFT key on the keypad. With SHIFT inactive, i.e. the SHIFT indicator not lit, the PROCESSING keys generate the words marked on them. If SHIFT is active, the keys generate the words marked *above* them.

As shown later, some of the words are always grouped together in control structures, whilst others are employed singly, such as PRINT which was used in the example of Section 2.5. Examples of the use of all the PROCESSING keys are given in the relevant Sections of this Chapter.

3.7 THE EDIT KEYS

Seven of the eight EDIT keys can also be used in conjunction with the DEFINITION keys and they have therefore been described earlier, in Chapter 9. However, for convenience, these seven keys, plus the eighth, are all described here. The EDIT keys act upon the information shown in the Display as explained below.

All the keys have an auto-repeat facility. If pressed and released quickly, only one step is performed, e.g., one character is deleted. However, if the key is held pressed, the relevant action will be repeated, e.g., characters will continue to be deleted, until either the key is released, or there are no characters left in the line. It is important to realise that holding the LINE DELETE key pressed for long enough will cause the entire program to be deleted.

To edit a processing program which is not currently being displayed, first press INIT, followed by the soft key EDIT. Enter the number of the line which is to be edited, then use the EDIT keys as described in the following sections.

Note that all new lines, or amended lines, must be transferred into program memory by pressing ENTER.

On completion of editing, pressing INIT followed by the soft key EXIT will produce the displayed state "Ready for command". Alternatively, the user may simply press the key for the next required function, e.g., RUN.

↑ STEP BACK ↓ STEP FORWARD	These keys enable the user to step through processing programs line by line in either direction, for inspection and/or alteration. If the Display is already showing the first or the last line, any attempt to step further will cause the Error Tone to sound.
←STEP LEFT → STEP RIGHT	These keys enable individual characters to be altered as follows. The ← key must be pressed first. This causes the cursor, in the form of a blinking rectangle, to appear superimposed on the extreme right-hand character in the line. Further operation of this key, and the → key if necessary, will align the cursor on top of the character to be changed.

When the new character key is pressed, the cursor will move one step to the right thus revealing the new character in place in the Display. There is no need to move the cursor clear of the Display before pressing ENTER.

CHARacter INSERT

To insert a character into the information in the Display, first press ← to produce the cursor. Position the cursor on the character which will *follow* the one to be inserted, and press CHAR INSERT. A gap will appear in the displayed information, in which the new character can be inserted.

CHARacter DELETE

To delete a character, first press ← to produce the cursor. Position the cursor on the superfluous character, press CHAR DELETE, the character will disappear and the remaining text close up to fill the gap.

LINE INSERT

The facility is intended primarily for programs written on consecutive lines. When a new line is to be inserted in such a program, there is no line number immediately available for it. Therefore the new line has to take over the number of the existing line which will ultimately follow it.

As a result, this existing line and all subsequent ones must have their line numbers incremented by 1. This renumbering, or resequencing, is performed automatically by the processor.

To insert a new line, first establish in the Display the existing line of program whose number is to be used for the new line. If the program is currently being edited, operate the ↑ or the ↓ keys until the required line number is shown. If the program is not currently being edited, press INIT followed by the soft key EDIT. The Display will now show:

"ENTER program line to begin edit".

Key in the required line number, and press ENTER.

Having established the required line number in the Display, press LINE INSERT, whereupon the Display will go blank, apart from the original line number. The new line information can now be entered.

The original line data, and all subsequent lines are automatically re-numbered to follow the inserted line.

Example: to insert the line PRINT AVERAGE to follow line 6010 in the section of program shown below.

```
6010 AVERAGE=SUM/5

6011 COUNTER=0

6012 ENDIF
```

Step 1: establish the line:

```
6011 COUNTER=0
```

in the Display.

Step 2: press LINE INSERT. The Display will now show just the line number 6011.

Step 3: key in PRINT AVERAGE and press ENTER.

The Display will now show:

```
6012 COUNTER=0
```

and if the program is Listed, it will appear as:

```
6010 AVERAGE=SUM/5

6011 PRINT AVERAGE

6012 COUNTER=0

6013 ENDIF
```

Resequence Failure. If the LINE INSERT operation is unsuccessful, see sub-section 3.1 "Entering Programs", under the heading "Use of Line Number 999" and sub-section 3.8 "Resequencing", for possible reasons.

LINE DELETE

To delete a line, operate the ↑ or the ↓ keys until the line to be deleted appears in the Display. Press LINE DELETE, the original line will disappear and the remaining lines automatically close up so that the next one in the sequence now appears in the Display. With processing programs, the line numbering is automatically adjusted.

Note: holding the key pressed causes lines to continue to be deleted until either the key is released or there are no further lines left.

3.8 RESEQUENCING

This facility is primarily intended for use where the program has been entered remotely via one of the interfaces, and is now to be edited from the Front Panel. Where programs have been written initially on non-consecutive lines, the processor can be instructed to resequence the line numbers such that they become consecutive. This is achieved by first putting the logger in "Ready for command" mode, then keying-in a line of the form:

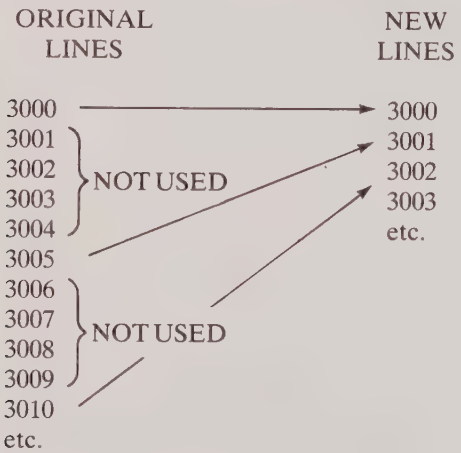
RE ␣ Old First Line Number, New First Line Number, Step Size.

Example:

The original program started on line 3000, using every fifth line number. Having completed the program, the writer wishes to eliminate all the gaps, i.e. have the program on consecutive lines, but still starting on line 3000. This would be keyed in as:

RE ␣ 3000,3000,1 ENTER

The line numbers will now be re-allocated as shown below:



The processor automatically initiates resequence whenever the LINE INSERT or LINE DELETE functions are used.

Error Code 161 indicates that an attempt to resequence has failed. This occurs if:

a. Line *x*999 is already in use and an attempt is made to add an extra line by using LINE INSERT.

b. Line number *x*999 would be exceeded by the resequence operation.

c. An attempt is made to transfer a program from one task to another, e.g. from Task 3 to Task 8 by:

RE ␣ 3000,8000,1

which is not permissible.

4 SYNTAX AND ERROR INFORMATION

All software languages have rules defining the form which the programming statements must take. These rules are called the "syntax". The syntax requires that commas, semi-colons, spaces or lack of spaces between characters, etc., are treated as essential parts of the statement, and not merely punctuation. Correct syntax is vital to the processor's understanding of the programs.

The general form for each main type of program statement, e.g. FOR . . . NEXT loops, is given in the relevant section. The programmer must follow these forms exactly, including all the spaces, commas etc., as shown, and not adding in any extra. Whilst it is not practicable to try and define specific rules for every possible line of program, as the acceptability of each new character in a line is largely determined by those preceding it, certain general rules can be stated as follows:

1. Spaces. These must be included in program lines only where specified.

a. **The PROCESSING keys.** The sixteen words provided by the PROCESSING keys are automatically preceded and followed by a space. If the user elects to create these words letter by letter from the keypad, which is perfectly permissible, the two spaces must be included.

b. **WRITE.** This word is similar to those provided by the PROCESSING keys, but has to be typed-in from the keypad. It must be preceded and followed by a space.

c. **Line Numbers.** Program line numbers are automatically followed by a space when they appear in the Display after pressing ENTER. Therefore the programmer should not add a space solely for the purpose of separating the line number from the rest of the line.

d. **Print Statement.** The word PRINT is always followed by a space, as described in rule 1a. When using a modifier with a Print Statement, the line, as typed, will appear in the Display as e.g., PRINT ␣ >. However, if the line is subsequently recalled to the Display, the processor will have "stripped" the space out, so that PRINT> will appear. The user need not attempt to re-insert the space.

e. **Expressions.** Expressions need not include spaces around the operator, e.g. type C1>4 not C1␣>␣4.

f. **Variables.** See rule 2b.

g. **Inverted Commas.** Spaces are perfectly acceptable anywhere within inverted commas.

h. **Minus Signs.** Except where the minus sign is used to indicate subtraction, or a negative exponent, it must be preceded by a space, e.g. a negative constant must be typed ␣−1.27E−6.

2. Variables

a. Within the limits on length specified in Section 6.6, the names of variables can consist of any mixture of alpha and numeric characters *except* that they must never begin with a number, e.g. TEMP4 is acceptable, 4TEMP is not.

b. The names of variables must not include "embedded" spaces, as the processor cannot recognise them. For example, C21 is acceptable, C ␣ 21 will be rejected and produce Error Code 159 "Illegal Variable".

3. Print Statements

a. Print Statements containing an operator and an indexed variable must never start with the variable, e.g. PRINT PRESSURE(I)*0.7 is wrong, PRINT 0.7*PRESSURE(I) is correct.

b. See also rule 1d regarding spaces.

4. Multi-Statement Lines

a. Multi-statement lines must include a colon as a separator between each pair of statements, as described in Section 3.3.

Error Checks
To prevent invalid statements corrupting the final program, three stages of checking are automatically performed, as follows:

1. As each new line of program is entered, it is checked for syntax errors *within the line.*

2. When RUN is pressed, the processor scans through the program *before* running it, to assign storage space for the variables etc. During this process, syntax errors which may be spread over several lines, e.g. in FOR . . . NEXT loops, are detected.

3. When the program is actually running, more deeply-buried errors are detected, e.g. that a Channel to be processed is not present in the associated Task.

Errors of the types described above are listed in Appendix 1, "Error Messages". Items 1 and 2 are classified as "Processing" errors, item 3 are "Run Time" errors.

5 CONSTANTS

The following types of constants are recognised by the processor:

Numeric
Numeric constants, of up to 6 significant figures can be entered, e.g.
123456 and 123.456

Decimal constants *must* have leading zeros included, e.g.
0.13 is correct.
.13 is wrong.

Exponent Form
The processor also recognises constants in exponent form, e.g. 1×10^6 can be entered as 1E6. Up to four significant figures can be entered, or output, in a format consisting of 10 characters maximum. For example, the positive constant, with positive exponent:

1.234E15

consists of 8 characters, whereas the negative constant, with negative exponent:

−1.234E−15

uses the full 10 characters.

Note Except where the minus sign is used to indicate subtraction or a negative exponent, it must be preceded by a space, e.g. in the example above, this must be typed in:

␣ −1.234E−15

The exponent format is automatically selected for output if the value is either:

a. Between, but not including, −0.01 and +0.01, i.e. is in the range −9.999E−3 and 9.999E−3 or:

b. Greater than 10^8 in either polarity. These values take the form: 1E8 or −1E8.

Messages

a. Text. To accompany a processed result, in Full format, e.g. "LABELLED RESULT". Can be up to 18 characters.

b. Units. To accompany a processed result, can be up to 3 characters, e.g. "RPM".

c. A message, such as, "SYSTEM OVERLOAD", can be up to 38 characters in Full format and up to 18 characters in Compact format. These messages are obtained by means of the PRINT, and WRITE, statements.

6 TYPES OF VARIABLE

Variables were introduced in Section 2, which included the following definition, repeated here for convenience:

"A variable is the name given to a location in the ORION memory in which a piece of information may be stored".

Section 2 also stated that variables can be classified under various descriptive titles. The programmer must be aware of the characteristics of each type, so that the most suitable may be chosen for a particular program. To put matters into perspective, the reader will encounter variables under *five* different names in this Chapter. Of these five, only *two* are completely distinct forms, these being:

1. Simple Variables

2. Indexed Variables

To these can be added:

3. Pre-defined Variables

which have already been introduced in Section 2. Pre-defined variables are a hybrid form which has been especially created for ORION *delta*. They possess certain attributes of both the two main forms.

The final two names to complete the list are:

4. User-created Variables

5. Global Variables

Of these two, the first has already been introduced in Section 2, and both "user-created" and "global" are merely adjectives describing certain properties possessed by both the simple and the indexed forms.

The salient features of all five categories are explained, in the order given above, in the following sections.

6.1 SIMPLE VARIABLES

Simple variables consist of only *one* element, that is to say, they can store only *one* piece of information. For example, the result from a single resistance thermometer could be processed, then stored in a user-created simple variable called TEMP. If two resistance thermometers are used, the programmer must create *two* simple variables, called, say, TEMP1 and TEMP2.

Thus, in a program employing only user-created variables, every piece of information to be stored must have a separate variable named for it. Simple variables are therefore used mainly when processing relatively few results, or where there are one or two different transducers amongst a host of otherwise identical ones, e.g. one resistance thermometer amongst several hundred strain gauges. The strain gauge results would be stored more efficiently in an indexed variable.

Simple variables can be introduced at any point in the program. Unlike the indexed variables described in the next Section, they must *not* be declared at the start of the program. The rules governing their names are given in Section 6.6.

6.2 INDEXED VARIABLES

Indexed variables consist of a number of elements, comprising an "array", which can be visualised as a honeycomb in which each cell, or element, stores one piece of information. Each element has its own, unique address, consisting of the variable name plus the element number. The rules governing the names are given in Section 6.6.

The general form of an indexed variable is:

```
VARIABLE(I)
```

where "VARIABLE" represents the name, e.g. STRAIN, and I represents an index number which has to be defined in the program for it to have any meaning.

The main purpose of the indexed variable is to allow the programmer access to large numbers of results in the shortest possible way, by referring to all the elements in an array in one brief statement. For comparison, consider a Task scanning five strain gauges. The results can be processed and stored in five simple variables called:

```
STRAIN1, STRAIN2 . . . STRAIN5.
```

These results can then be output by means of five Print Statements, (described in Section 9.3), as follows:

```
7009  PRINT STRAIN1
7010  PRINT STRAIN2
.
.
7013  PRINT STRAIN5
```

It can be seen that this type of program would become very lengthy if the number of strain gauges were increased to, say, 99.

However, by using an *indexed* variable, such as:

```
STRAIN(I)
```

and defining I in the program as covering the range 1 to 99, the statement:

```
7009  PRINT  STRAIN(I)
```

will cause the processor sequentially to substitute the values 1, 2, 3 etc. up to 99, for I, and hence print out the 99 results from the variables STRAIN(1) to STRAIN(99). Note that, as explained in Section 6.5 "Global Variables", the index need not always be represented by I.

Indexed variables are mainly employed in FOR . . . NEXT loops, which are described in Section 8.2

Each element in an array can be addressed individually as, e.g. A(1), A(2) etc. The presence of the brackets is mandatory, the processor recognises that A(1) and A1 are totally dissimilar; the first is to be treated as part of an indexed variable, the second as a simple variable.

Declaring Indexed Variables

The name and number of elements in each indexed variable must be declared at the start of the program. This notifies the processor to reserve the required number of locations in the memory. A typical declaration in line 1000 is:

```
1000  MEAN;5      (note the semi-colon separator)
```

This declaration notifies the processor that there is an indexed variable called MEAN, possessing 5 separate elements, called MEAN(1), MEAN(2) etc.

Each declaration is effectively a statement. Therefore, several declarations can be included in the same line, provided that the rules governing multi-statement lines are observed, as given in Sections 3.2 and 3.3.

As described in Section 6.5 "Global Variables", the same variables can be shared by several Tasks. When an indexed variable is shared in this way, it must be declared *only* in the lowest numbered Task which uses it.

Declarations can be extended to include additional information such as units, explanatory text etc., as described in Section 9.5 "Output Format Declaration".

A special group of indexed variables, called the Scanned Channel functions, are described in Section 13 "Random Channel Access".

6.3 PRE-DEFINED VARIABLES

Pre-defined variables were introduced in Section 2, where it was explained that their names have been permanently reserved for them in the ORION memory. These variables have been specially developed for ORION, and they share certain properties of both the simple and the indexed forms. Strictly speaking, they constitute *three* indexed variables, named C, H, and L. Each of these variables can contain up to 600 elements to conform to the maximum 600-Channel capacity of the logger.

By the rules of indexed variables, it would seem that each element should always be referred to in the program as, e.g. C(1), H(50), L(300) etc. However, these variables are used very frequently, particularly the C1 to C600 group in which the unprocessed Channel results are originally stored. Therefore, for the users benefit, the processor has been permanently programmed to recognise them in the simple variable form of, e.g. C1, H50, L300 etc., i.e. the programmer may omit the brackets round the element number.

Also, unlike other indexed variables, the pre-defined variables do not have to be declared at the start of the program. However, the normal indexed form can be used in programs in, say, a FOR . . . NEXT loop, as mentioned in the previous section "Indexed Variables". For example, the following section of program:

```
7054  FOR I=1 TO 200
7055  PRINT C(I)
7056  NEXT
```

will cause the results of the first 200 Channels to be output. (FOR . . . NEXT loops are described in Section 8.2). H(I) and L(I) are also recognised by the processor, as the High and Low limits respectively of Channel(I).

The features described above are common to all three groups of pre-defined variables. However, the variables C1 to C600 also differ in certain respects from the variables H1 to H600 and L1 to L600, as shown in the two following sub-sections.

Variables C1 to C600

1. The programmer has no control over the storing of the original results in these variables, as this is performed by the processor as the Channels are scanned. Each result is automatically placed in the correct location, e.g. Channel 6 result is always stored in the pre-defined variable C6. However, the programmer does have a choice when processing these results, as to whether the intermediate, or fully-processed, versions shall be stored back in the pre-defined variables, thus over-writing the original result, or in user-created variables. The factors which may influence this decision are discussed in Section 6.7.

2. The Channel results cannot be processed as part of a Task unless they have already been declared in that Task, as explained in Section 2, Rule 2.

3. These variables are *not* global. Each one is accessible only to the program contained in the Task which scanned the Channel.

4. The processor recognises the names C1 to C600 in all the 8 Tasks, but this does not mean that there are 8 x 600 locations permanently available. There is a shared "pool" of buffers which act as variables for storing the Channel results. As each result is obtained, a buffer in the pool is assigned the relevant name, e.g. C1, and the result is stored in it.

The buffer retains the identity C1 only until processing is complete, whereupon it becomes available for re-use. For the majority of applications, results are processed fast enough to keep a reserve of unassigned buffers in the pool. However, in exceptional circumstances, e.g. a fast scanning rate coupled with extensive processing, all the buffers may have been filled. In this case, the scanning rate of the logger automatically slows down until buffers become available.

5. Any results stored in these variables, whether processed or not, are over-written on the next scan, i.e. the original values are replaced by the new ones.

6. The results stored in these variables, whether processed or not, can be output as shown in Fig. 12.6.

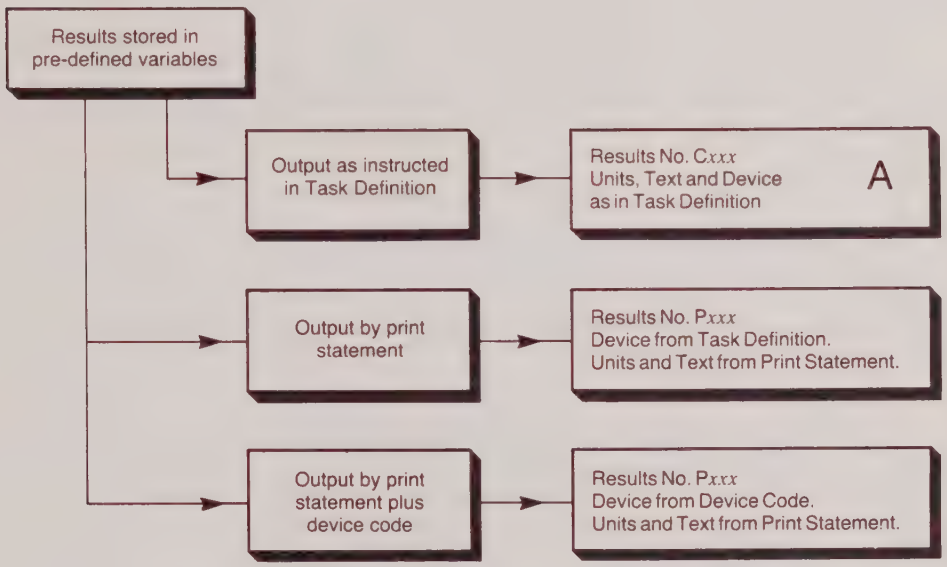

Results stored in pre-defined variables
Output as instructed in Task Definition
Results No. Cxxx
Units, Text and Device as in Task Definition
A
Output by print statement
Results No. Pxxx
Device from Task Definition.
Units and Text from Print Statement.
Output by print statement plus device code
Results No. Pxxx
Device from Device Code.
Units and Text from Print Statement.

Fig 12.6 *Outputting Results from Pre-defined Variables*

Notes: 1. Print Statements, device codes and Extensions are described in Sections 9.3 and 9.4.

2. When the Print Statement is used, with no additional processing such as PRINT C1*10, it merely copies the result that is printed in "A" in Fig. 12.6 above.

3. If result "A" is not required, the KILL statement, described in Section 9.9 may be used.

Variables H1 to H600 and L1 to L600

1. The values stored in these variables are used as High and Low alarm limits on the Channel results, e.g. H1 and L1 are the High and Low limits for Channel 1, H2 and L2 for Channel 2 etc. The values are usually entered initially during Channel Definition, as shown in Chapter 9.

2. The values entered in the variables are read by the processor as soon as RUN is pressed but before scanning commences. When the logger is halted, the values can be changed by entering new ones in the Channel Definition.

3. Unlike C1 to C600, the contents of these variables are not over-written on each scan.

4. The values can also be changed by means of a program statement. For example, the High Limit on Channel 12 may have been set to 7.9 by the Channel Definition. The program could then include a statement to warn the user when Channel 12's result exceeds 7.9, but as this may not be dangerous, the limit could then be automatically increased to 8.9 ready for the next scan. Using an IF . . . THEN statement, as explained in Section 8.1, the program could include the lines:

```
5111 IF C12>8 THEN
5112 H12=8.9
```

5. These variables are *global*. Therefore when the lines shown in item 4 are executed, the High limit on Channel 12 is increased to 8.9 for *all* Tasks that scan Channel 12.

6. Limits which are unique to a particular Task can effectively be created by using the IF . . . THEN statement mentioned in item 4 above. For example, Task 3's program could contain the line:

```
3330 IF C16>3.14 THEN
```

   followed by further lines detailing the appropriate action, e.g. show a warning message in the Display, or Print an alarm message, as explained in Section 9 "Output Statements".

   This type of alarm limit does not affect the results from Channel 16 in any of the other Tasks because it has not changed H16.

7. As shown in the block schematic, Fig. 12.1, these variables are the only ones to be stored in a special area of memory, called the data-base. Because of this, their contents exist only as numbers, with no text, units etc.

8. Although, as explained at the beginning of Section 6.3, they are indexed variables, nevertheless care must be taken if they are used for storing anything other than High or Low limits. This is because, unlike other ORION variables, they are located in the data base. Therefore, the amount of space allocated to them varies according to the type of Input/Output modules fitted.

9. A logger which is configured for n Channels, automatically has the requisite n H and L locations reserved in the data base. If the configuration is subsequently changed to n + m or n − m Channels, the H and L locations are automatically altered accordingly.

6.4 USER-CREATED VARIABLES

The behaviour of user-created variables is described in Section 2, which also explains the basic difference between these variables and the pre-defined ones. A further definition can now be added, namely that *all* the variables that ORION can recognise are *user-created*, with the sole exception of the pre-defined ones. Hence, whether the variable is simple or indexed, it can be created only by the user giving it a name in the program. The rules governing this naming are given in Section 6.6.

As long as a program remains in memory, any user-created variables named in it will be re-created by the processor each time that RUN is pressed to start scanning. This re-creation process occurs for *every* program stored in the memory, even if "Select end-of-scan processing" is "Off" for any or all of the Tasks. This has the advantage that any indexed global variables, that are declared in a program which is switched "Off", will still be created in memory by the processor. Hence, these variables are still available for use by programs in other Tasks, without having to be declared again therein.

6.5 GLOBAL VARIABLES

A variable is said to be "global" when it is accessible from any Task rather than just the one in whose program it first appears. All the variables that ORION can recognise are global, with the sole, but very important, exception of the pre-defined variables C1 to C600. As explained in Rule 1 of Section 2, each Channel result, such as C1, is only accessible within the Task that scans it.

Note that H1 to H600 and L1 to L600, although also pre-defined, are nevertheless global. The user must take care when using global variables that a statement in one Task does not inadvertently affect the performance of other Tasks. For example, the simple variable MEAN, which is global, may have been used in Task 1 to store the value of the expression:

(C1 + C2 + C3)/3

Later on, having forgotten its use in the still-functional Task 1, the programmer may use MEAN again in Task 8, to store the value (C8 + C9 + C10)/3. The processor is now asked to store two totally different results in the same variable! It solves this problem by over-writing whatever is in MEAN with the result derived from the latest Task performed.

This may be fine for that Task but, because of the multi-tasking nature of ORION, it may yield a totally invalid result for the other one. Also, if such an inadvertently-shared variable is used for totalising over several scans, it will give a misleading result in both Tasks. The processor cannot identify the presence of a shared variable as a syntax error, as it may have been introduced deliberately as a perfectly valid programming technique. To guard against such confusion, the programmer could form the habit of using the Task number as a suffix for all variables contained in it, e.g. MEAN1 for Task 1, MEAN5 for Task 5 etc.

The above hazard must be watched for particularly when using the traditional text-book letter I to represent an index. I is used throughout this Chapter to represent the index in the general form of various statements, such as indexed variables. For example, TEMP(I), where the number of elements represented by I must be declared in the program as, e.g., TEMP;5.

Later on, this variable will probably be included in a FOR . . . NEXT loop, as shown below, where all five elements are to be output. (FOR . . . NEXT loops are described in Section 8.2).

```
FOR I=1 TO 5
PRINT TEMP(I)
NEXT
```

where I can be included in the program exactly as written here.

Note, however, that I is a *global variable*, so can be used in only *one* FOR...NEXT loop in *one* Task. Again, this problem can be overcome by using suffices, e.g. I1, I2 etc., or even A, B, D etc., instead of I. See Section 8.2 "FOR . . . NEXT Loops", under "Loop Variables".

6.6 RULES GOVERNING VARIABLE NAMES

Pre-defined variables already have their names allocated to them, the only choice open to the programmer being whether to use them as simple variables, in the form C1, H1, L1 etc., or in the indexed form C(I), H(I), L(I) etc. The factors influencing this choice are given in Sections 6.1 and 6.2, where the properties of simple and indexed variables are described. With the above exception, the following rules apply to all simple and indexed variables.

Some of the rules are also included in Section 4 "Syntax".

1. Names can consist of a combination of up to eight letters and numbers, but must *never* begin with a number.

2. Names can range from a single letter, e.g. A, B, D etc., to a complete word such as AVERAGE.

3. The letters C, H and L are reserved for the pre-defined variables and therefore must *not* be assigned to user-created variables. Any attempt to do so will produce Error Code 159 "Illegal Variable Name".

4. Identity numbers may be included as part of the name of simple variables, as in A1, A2, AVERAGE1, AVERAGE2 etc., and of indexed variables, as in A1(I), AVERAGE6(I) etc., provided that the total number of characters does not exceed eight.

   Note that the index part, shown here in the general form (I), does not count towards the eight permitted characters.

5. Names must not include embedded spaces, as shown in the following examples:

   a. C21 not C ␣ 21

   b. MINTEMP not MIN ␣ TEMP

   c. AVERAGE6 not AVERAGE ␣ 6

   d. PRESSURE(3) not PRESSURE ␣ (3)

The example programs given throughout this Chapter show the correct format of various types of names.

6.7 APPLICATIONS OF PRE-DEFINED AND USER-CREATED VARIABLES

The following list of features is included to assist the programmer in deciding which of these two categories of variable is best suited to a particular application. The comments on pre-defined variables apply whether the results stored therein have been processed or not.

1. Making Channel results global

A result contained in one of the pre-defined variables C1 to C600 is accessible only to a program contained in the Task that originally scanned it. However, if the result is transferred from the pre-defined variable to a user-created variable, with or without any other processing, e.g. by writing A=C1, this new variable, A, is global, and therefore accessible to *all* Tasks.

2. Totalising over several scans

Results stored in pre-defined variables are over-written on each scan. Therefore these variables cannot be used for totalising over several scans. User-created variables, although cleared when RUN is pressed to start scanning; can be used to accumulate results over all the scans that occur until RUN is pressed again to halt the logger.

3. Selecting the output device

3.1 Pre-defined Variables

a. Results are *always* sent to the output device specified in the Task Definition.

b. Results can be *copied* to a different device by means of a Print Statement which includes the code for that device, as explained in Section 9.3.

3.2 User-created Variables

Results stored in user-created variables cannot be sent to an output device at all until they have been included in a Print Statement, which may take one of the following two forms:

a. A Print Statement with *no* device code included; this will simply send the results to the output device specified in the Task Definition.

b. A Print Statement which includes a device code; this will send the results to the device represented by that code. For a full explanation, see Section 9.3 "Print Statements".

4. Information accompanying the output result

4.1 Pre-defined Variables

Results stored in pre-defined variables can be accompanied by the following information:

a. As stated in 3.1a above, results are always sent to the output device specified in the Task Definition, and are accompanied by any text and units that are included in the Channel Definition.

b. If a result is also included in a Print Statement, with *no* device code and *no* Extensions, it will be copied to the device specified in the Task Definition. Therefore two identical results will appear viz:

1. The result from the pre-defined variable, as in "a" above.

2. A duplicate result produced by the Print Statement, but with *no* accompanying text etc. Hence, there is seldom any point in using this method.

c. If a result is included in a Print Statement with a device code but *no* Extension, it will be copied to the specified device with *no* additional information.

d. If a result is included in a Print Statement, with a device code *and* Extensions, it will be copied to the specified device accompanied by the information included in the Print Statement Extensions, as described in Section 9.4.

4.2 User-created variables

Results stored in user-created variables fall into two categories, both of which require a Print Statement in order to appear as output results at all:

a. Simple variables. If text, unique processed results number, and units information are required to accompany the results, they must be included in Print Statement Extensions, as described in Section 9.4.

b. Indexed variables. Information, as described above for simple variables, can also accompany results stored in indexed variables. However, there are now two possible sources of this information, i.e. Print Statement Extensions and Output Format Declarations. Every program which introduces an indexed variable must declare it at the top of the program. This declaration can be extended to include text, units etc., whereupon it becomes an Output Format Declaration. Also, to send results to an output device, the program must contain a Print Statement, which can similarly be extended to include text, units etc. The programmer should choose which to use after studying the details of both forms in Sections 9.4 and 9.5.

5. Economising on memory and cartridge space

Where space is at a premium, and the original Channel results are no longer required, savings can be effected by putting the processed results in the pre-defined variables C1 to C600. For example, consider the case where Channel 1 is measuring the current supplied to a load, and Channel 2 is measuring the voltage across it. If the user is interested only in the power dissipation V × I, the original results may be discarded.

Two ways of achieving this are:

a. By writing the line:

```
2020  C1=C1*C2
```

This automatically puts the power result in the variable C1, erasing the original current value. To erase the voltage value in C2, the next line could contain a KILL statement, as described in Section 9.9:

```
2021  KILL 2
```

The storage space previously occupied by the result in C2 is immediately made available for re-use, instead of being retained until the results are output. This helps to maintain the scanning rate when large numbers of Channels are being scanned, as stated in Chapter 1 when referring to the measurement buffer.

This section of the program is now complete; the power results stored in C1 will be output at the end of the program, in accordance with the instructions given in the Task Definition. The result will also be accompanied by any text and units entered in the Channel Definition. In this case, looking ahead, the programmer should enter W (for Watts) rather than A (for Amperes), as the units.

b. Alternatively, the programmer could introduce a user-created variable, called A, in the line:

```
2020  A=C1*C2
```

"A" now stores the power value, and the current and voltage values remain in C1 and C2. A further line could employ the KILL statement, as before:

```
2021  KILL 1,2
```

This immediately releases the storage space of the two pre-defined variables C1 and C2. A third line must now be added to output the result:

```
2022  PRINT A
```

This line can be augmented by a Print Statement Extension to include text, units etc., to accompany the processed result, as described in Section 9.4.

Comparison. Comparing these two methods, it can be seen that method "a" employs only two lines of program compared to three for method "b". Also, as stated in Section 16, when using Coded format for recording, the user-created variables occupy *two* records where pre-defined variables occupy only one, which obviously affects the number of results that can be stored on the cartridge.

A third method where, say, Channel 3 is not being used, but has been declared in the Task Definition is to write:

```
C3 = C1*C2
```

This confers all the space-saving benefits of using pre-defined variables, but also enables the user to retain the original current and voltage readings.

6. Processing Speed: Checking results against limits

Pre-defined variables are accessible to the high-speed alarm limit checking set up in the Channel Definition. This method is faster than storing the results in user-created variables, then checking them in a series of IF . . . THEN statements. This latter method has its uses, however, for avoiding the global effect of changing the pre-defined variables H and L, as shown in Section 6.3.

7 MATHEMATICAL OPERATORS AND FUNCTIONS

7.1 MATHEMATICAL OPERATORS AND PRIORITIES

The following groups of operators are available for processing ORION *delta* data, using the symbols shown.

Mathematical Operators

+ Addition,	i.e. A+B	write as A+B
− Subtraction,	i.e. A−B	write as A−B
* Multiplication,	i.e. A×B	write as A*B
/ Division,	i.e. A÷B	write as A/B

( )Parentheses

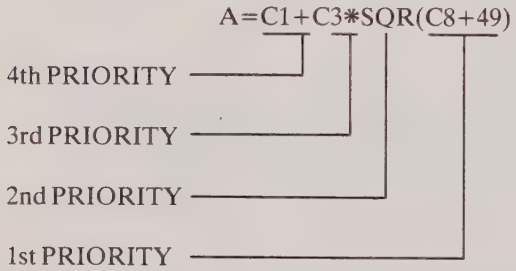
** Integer Exponentiation see page 12.33

The operations are performed according to standard mathematical rules, i.e.:

1st Priority	:	Parentheses
2nd Priority	:	Integer Exponentiation
3rd Priority	:	Division
4th Priority	:	Multiplication
5th Priority	:	Addition
6th Priority	:	Subtraction

Pending Operations

Expressions containing a maximum of 4 pending operations can be entered. The expressions are evaluated from left to right, and as the processor finds higher priority calculations to perform, the lower priority ones are left pending, e.g.



If a fifth operation is added to the above expression, e.g.

A=C1+C3*SQR(SIN(C8)+49)

the logger will reject the expression and indicate an error.

Parentheses

Parentheses can be nested up to 4 levels, e.g. A=LN(SQR((5*(C4+2.3))/13))

1st LEVEL = $5\times(C4+2.3)$

2nd LEVEL = $\frac{5\times(C4+2.3)}{13}$

3rd LEVEL = $\sqrt{\frac{5\times(C4+2.3)}{13}}$

4th LEVEL = $\log_e \sqrt{\frac{5\times(C4+2.3)}{13}}$

Integer Exponentiation

** e.g. for A^2 type A**2

Note that the exponent must be an integer, i.e. it is not permissible to enter expressions of the type:

8 ** 0.25

for the fourth root of 8. Non integer exponents may be calculated by using logarithms.

7.2 TRIGONOMETRIC AND EXPONENTIAL FUNCTIONS

ABS(X) Absolute value of x, e.g. +6.6 becomes 6.6
−6.6 becomes 6.6

SQR(X) Square root of x. x must be positive.

SIN(X)
COS(X) } Trigonometric functions, x must be in radians.

ATAN(X) Inverse trigonometric function, i.e. arctan x, where x is in radians.

EXP(X) e^x, i.e. e raised to the power of x.

LN(X) $\text{Log}_e x$, i.e., the natural logarithm of x.

INT(X) Integer part of x, e.g. +6.6 becomes 6.
but −6.6 becomes −7.

This is because ORION takes the first whole number *less than* the value given. Mathematically, negative numbers are considered to be less than zero; therefore −7 is less than −6.

These functions are simply written into the program in the general form:

```
VARIABLE=FUNCTION(ARGUMENT)
```

e.g. with the name RESULT assigned to the variable, a typical program line could be:

```
4020 RESULT=SQR(C6)
```

When this is printed out, the value stored in the variable RESULT will be the square root of the result from Channel 6, as measured in Task 4.

A more complex function would be:

```
4020 RESULT=SQR(C6*7)
```

which will yield the square root of (7 times the result from Channel 6). This complexity can be extended up to 4 levels of expression nesting in any one statement. Expressions which require more than 4 nesting levels must be divided into two or more statements.

7.3 RELATIONAL AND BOOLEAN OPERATORS

Relational Operators

<	Less than,	e.g. A<B means A less than B
>	Greater than,	e.g. A>B means A greater than B
=	Equal to,	e.g. A=B means A equals B
<>	Not equal to,	e.g. A<>B means A does not equal B
<=	Less than or equal to	
>=	Greater than or equal to	

Boolean Operators

Two of these are available:

AND, OR } These operators can be used in statements of the type:

IF A>P AND B>Q THEN . . . . . .

IF A>P OR B>Q THEN . . . . . .

8 CONTROL STRUCTURES

8.1 IF . . . THEN STATEMENTS

The processor can be instructed to choose between two different program paths, depending upon the way in which certain prescribed conditions have been met. The point in the program at which the decision is taken is called a branch. The IF . . . THEN statement provides such a branch, the simplest form being:

IF *expression* THEN *statement*:ENDIF

An example of this is:

```
4008 IF C1>5 THEN
4009 TRIGGER 6
4010 ENDIF
4011 . . . etc.
```

Here, line 4008 contains the expression C1>5, whereby the value of Channel 1 is compared with the constant 5. This comparison yields one of two results:

a. TRUE, meaning that Channel 1 value *is* greater than 5. In this case, the statement in the following line, i.e. TRIGGER 6, is actioned, causing Task 6 to run concurrently with the original Task. The processor now automatically proceeds to the next line, i.e. 4010, which contains the instruction ENDIF. This instruction tells the processor that the IF . . . THEN statement is now complete, and therefore it can action line 4011.

b. FALSE, meaning that Channel 1 value is less than, or equal to, 5. In this case, the processor ignores the statement in line 4009 and seeks the first available ENDIF, which it finds on line 4010. The processor then proceeds to action line 4011.

Multiple Statements

Although the preceding example contains only one statement between THEN and ENDIF, multiple statements are permissible. Subject to the restrictions discussed in sub-section 3.2. these statements can be included on the same line, but in the example below they are shown on separate lines.

```
4008 IF C1>5 THEN
4009 TRIGGER 6
4010 ABORT 5
4011 PRINT "TEMP>LIMIT"
4012 ENDIF
```

As before, a TRUE result causes the three statements in lines 4009, 4010 and 4011 to be actioned, followed by ENDIF in line 4012. A FALSE result causes the processor to ignore all three statements and move straight to line 4012 to execute the ENDIF statement.

ELSE

The previous examples simply offer the processor the choice of either executing the "THEN" statements, or ignoring them and proceeding directly to the ENDIF statement. By including ELSE in the structure, a further choice of statements can be offered. Whenever the expression is FALSE, the processor will seek either an ELSE *or* an ENDIF; whichever is found first will be actioned.

The simplest form is:

IF *expression* THEN *statement 1*:ELSE *statement 2*:ENDIF

Extending the previous single-statement example:

```
4008 IF C1>5 THEN
4009 TRIGGER 6
4010 ELSE
4011 PRINT "FREEZER O.K."
4012 ENDIF
```

A TRUE result causes the processor to action line 4009, as described in the previous example but now it also nullifies the ELSE option. Hence, as before, the processor jumps down the program to execute the final ENDIF statement.

However, a FALSE result causes the ELSE option to be actioned, i.e. line 4011 PRINT "FREEZER O.K." is executed, followed by the final ENDIF statement. The ELSE option can also contain more than one statement.

Nesting

IF . . . THEN structures are nestable up to 4 levels, as shown in the following example:

```
1006  POWER=C1*C2
1007  IF POWER>5 THEN                                  Level 1
1008  PRINT "POWER>5"
1009  ELSE
1010  IF POWER>6 THEN                                  Level 2
1011  PRINT "POWER>6"
1012  ELSE
1013  IF POWER>7 THEN                                  Level 3
1014  PRINT "POWER>7"
1015  ELSE
1016  IF POWER>8 THEN                                  Level 4
1017  PRINT "POWER>8"
1018  ELSE
1019  PRINT "POWER<5"
1020  ENDIF : ENDIF : ENDIF : ENDIF
```

Note that there must be one ENDIF statement for every IF and that they are nested within each other, i.e. the final IF statement nests with the first ENDIF statement etc. In the example above:

the IF in line 1016 nests with the first ENDIF statement on line 1020

the IF in line 1013 nests with the second ENDIF statement on line 1020

the IF in line 1010 nests with the third ENDIF statement on line 1020, and so on.

Further statements may exist between ENDIF statements, and will be actioned in the appropriate nest.

8.2 FOR . . . NEXT LOOPS

These structures define the beginning and end of a program loop. A loop is a set of instructions which are repeated over and over again, each time being modified in some way until a terminal condition is reached. The simplified general form is:

FOR I=*x* TO *y*, *carry out instructions*, NEXT

Here the loop variable I is first set to the value *x* and the instructions are carried out. NEXT causes I to be set to (*x*+1) and the same instructions are repeated. The loop is repeated until I has been set to *y* and the instructions are carried out for the last time.

A simple application of this structure would be to print the results from the first ten Channels. This could be achieved, a Channel at a time, by:

```
1000 PRINT C1
1001 PRINT C2
1002 PRINT C3
```

etc., but this would become very unwieldy for a large number of Channels. A more elegant solution is:

```
1000 FOR I=1 TO 10
1001 PRINT C(I)
1002 NEXT
```

Here, the logger will sequentially substitute in line 1001 the values specified for I in line 1000, hence generating all the Channel numbers from 1 to 10 inclusive, and causing the values to be printed. Each time that the program reaches the word NEXT in line 1002, it examines the value currently assigned to I in line 1001, and if this is within the specified range of values, it will increment again by +1.

On the final repeat of the loop, the value of the 10th Channel is printed. I now equals 10, so the following word NEXT is ignored and the program continues with line 1003, if it exists, or terminates if it does not.

Loop Variables

Note that the loop variable named I in the previous example is also a *global* variable, and therefore whatever values are assigned to I in one Task will apply if I is used in any other Task. There are several solutions to this problem:

1. Use the simple variable I1 as the first loop variable required, I2 as the second, etc. If there is only one FOR . . . NEXT loop in each task, then I1 could be used for Task 1, I2 for Task 2, etc., although this sequence is not mandatory.

2. Use a different letter for each new loop variable, but do not use C, H and L, which are reserved for pre-defined variables, as explained in Section 2.4.

3. Assign names to the loop variables, up to eight characters long, e.g. LOOP1, LOOP2, etc.

Note Having declared a loop variable, a fixed value cannot be assigned to it (e.g. I=72) in its own Task or any other Task.

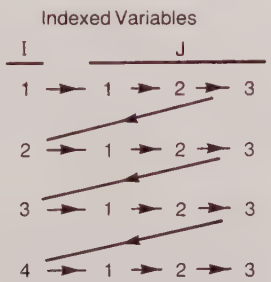
Nesting FOR . . . NEXT loops are nestable up to 4 levels, but for simplicity only 2 levels are shown in the example below:

```
1008  FOR I=1 TO 4                    Level 1
1009  FOR J=1 TO 3                    Level 2
1010  statement
1011  NEXT
1012  NEXT
```

There must always be one NEXT statement for every FOR statement, and the resulting loops nest within each other. In the above program:

the FOR statement in line 1009 loops with the NEXT statement in line 1011 and:
the FOR statement in line 1008 loops with the NEXT statement in line 1012

The above loop will be repeated 4 times, successively assigning values to I and J as shown in the following table.



Further statements could be added between the two NEXT statements, in which case they would be actioned 4 times in the above program example.

STEP

FOR . . . NEXT loops can be made more flexible by including a facility called STEP. This is included in the general form as:

FOR I=*x* TO *y* STEP Δ, *carry out instructions*, NEXT

The STEP value must be an integer, which may be positive or negative, but need not be declared if it is +1 as this is the default value. STEP is taken to be +1 in all the preceding examples.

If the simple example given earlier is modified as follows:

```
1000  FOR I=1 TO 10  STEP 2
1001  PRINT C(I)
1002  NEXT
```

Alternate channels only will be printed, viz 1, 3, 5, 7 and 9. With STEP=3, Channels 1, 4, 7 and 10 will be printed etc.

The sign for the value of STEP must agree with the direction of change of I, i.e. for counting down, STEP must be negative as in:

FOR I=10 TO 1 STEP −1

8.3 GOTO

This word is used in the general expression GOTO *Tnnn*, and means: "Continue execution of the program at line *Tnnn*"

where *T* = the number of the Task which contains the program and
nnn = line number in the same Task's program, e.g. a program in Task 1 must not contain the statement: GOTO 2301, as this line is in Task 2.

GOTO is used to transfer execution of the program forwards or backwards, ignoring all the lines that are skipped. If transferring backwards, the user should beware of inadvertently creating an endless loop. GOTO must not be used to exit from IF . . . THEN and FOR . . . NEXT structures.

Line numbers involved in GOTO operations are automatically adjusted during resequencing, e.g. following the use of the LINE INSERT key.

If written on a multi-statement line, GOTO must be the last statement on that line.

9 OUTPUT STATEMENTS

9.1 INTRODUCTION

Results are sent to output devices in a queue. Those from the pre-defined variables, as directed from the Task Definition, always appear first whether processed or not. They are followed by any results from the user-created variables. These latter can appear in the queue only if placed there by a Print Statement in the program.

Results added to the queue by the Print Statement *always* appear between the Start and Done Task markers. The command PRINT actually means:

"send to the device specified"

not just to the printer. The format is determined by the Task Definition entry. Print Statements are described in Section 9.3 and their associated Extensions in Section 9.4.

Additional information to accompany results from indexed variables can be included in an Output Format Declaration, as described in Section 9.5. The relative advantages of Output Format Declarations and Print Statement Extensions are described in Section 9.7.

Processed result numbers can be manipulated in various ways, as described in Section 9.6.

The programmer can also use a Write Statement, as described in Section 9.8. This Statement is not involved with the queue of output results, but causes a message to appear on the selected output device as soon as the processor decodes it in the program. This message may contain a calculation, based on results stored in the variables, but does *not* appear within the Start and Done Task markers.

There are two other forms of Output Statement:

a. KILL, which is used to delete redundant Channel results from the output information. This statement is described in Section 9.9.

b. SET . . . TO. This statement is used to control the logger's analogue and digital outputs, and is described in Section 9.10.

9.2 OUTPUT FORMATTING

Data handling within ORION is greatly simplified by what is called "formatting", whereby specific items of information must always be placed in the same position, or "field", in a sequence. Then, once the processor has recognised that a particular sequence is to be actioned, it knows exactly where to look within it for any specific piece of information. Chapter 11 shows several examples of this formatting, which is all performed automatically by the logger; all that the user has to do is to ensure that the individual items of information are entered in the correct fields.

In the Channel and Task Definitions, much of the work has already been done. The menus are presented in such a way that, as each item is selected, the relevant data is automatically stored in the correct field within the memory. Hence, when the results stored in the *pre-defined variables* are sent to an output device, the processor can refer to the relevant fields in the data base for any additional information, such as text and units, that should accompany them.

However, results to be output from *user-created variables* need to have had their additional information supplied, as part of the processing program, in some form of output statement. In the sections that follow, the user is shown the format in which the various output statements must be made.

9.3 PRINT STATEMENTS

The action of the Print Statement is to add a processed result to follow the scanned Channel results. This processed result will appear within the confines of the Start and Done Task markers. Note that the Print Statement may send results to *any* selected output device, not just the one specified in the Task Definition.

The Print Statement can be extended to make the output results more meaningful, as described in Section 9.4.

The general form of the complete Print Statement, with Extensions, is:

PRINT [*modifier*] ␣ [*device code*] ␣ *item*;[*extensions*],*item*;[*extensions*], . . . etc.

The square brackets denote options and are *not* included in the actual program. This Section describes the Print Statement proper, i.e.:

PRINT [*modifier*] ␣ [*device code*] ␣ *item*

Items
The simplest form of the Print Statement is:

PRINT *item*

There are three distinct forms, as follows:

a. **PRINT *expression*.** Expressions are defined in Section 3.4; an example is the simple variable TEMP, so the Statement, as given in the program, becomes:

```
2054 PRINT TEMP
```

This Statement causes the result stored in TEMP to be sent to the output device specified in the Task Definition. The result will have no text or units, and will be labelled as processed result number 1, i.e. P001. Text and units can be added, and the result number changed, by adding Extensions, as described in Section 9.4.

Example: Task 1 is scanning a d.c. voltage on Channel 1, and logging all results in full format. A PRINT *expression* statement can be included anywhere in the processing program, and could introduce a variable called e.g. NORMAL.

The program could be:

1080 NORMAL=2 (which defines the NORMAL value as 2)

1081 PRINT NORMAL−C1 (which calculates and prints any difference between the value of C1 and the NORMAL value).

This program will extend the usual print-out to give:

```
TASK TITLE
ST1    12:24:56.7
:
C001   1.2345 VDC          value of C1
:
P001   0.7655   .          processed result

DT1
```

Note that, whilst an expression may include an indexed variable, a PRINT *expression* must *never* begin with one. Thus:

2030 PRINT PRESSURE(I)*0.7 is forbidden

2030 PRINT 0.7*PRESSURE(I) is permissible

b. **PRINT *indexed variable*.** This Statement causes results to be output as for "a" above, except that the result from each element is automatically numbered sequentially, starting from 1. For example, if the variable declaration:

1000 DEPTH;3

is followed by:

1010 PRINT DEPTH(I)

this will produce a print-out of the form:

P001 *result from DEPTH(1)*

P002 *result from DEPTH(2)*

P003 *result from DEPTH(3)*

Text, units and additional numbering information can be added either by Extensions or by an Output Format Declaration, as explained in Section 9.5.

c. **PRINT "*string*".** "String" represents a sequence of alpha-numeric characters, comprising a text message which must be enclosed in inverted commas. The statement causes the "string" text to be inserted in, for example, the print-out of a scan, *before* any results but *after* any other text that precedes the "string" in the program. Up to 18 characters may be used for compact format and up to 38 for full format output.

Example: With Task 3 scanning Channels 1 to 5 and storing processed results in a variable called DEPTH(I). The program could start with the declaration of the variable, i.e.

3000 DEPTH;5

Later in the program, after the results from the pre-defined variables C1 to C5 have been processed and stored in the user-created variables DEPTH(1) to DEPTH(5), all results are to be output. The program continues:

3098 PRINT "ALL TESTS PASSED" This is a PRINT "*string*" statement.

3099 FOR I=1 TO 5

3100 PRINT DEPTH(I)

3101 NEXT

The print-out will be of the form:

WATER LEVELS	This is the Task Title, as specified in the Task Definition. It always takes precedence at the top of the print-out.
ST3 4:54:36.2	Start Task number and time of start.

* ALL TESTS PASSED This is the "string" from line 3098. There is no ambiguity as to its position, as it constitutes the only PRINT "*string*" statement in the program. The asterisk is added automatically by the logger.

: 1st unprocessed result's text, as given in the Channel Definition.

C001 unprocessed result from C1.
C002 unprocessed result from C2.
C003 unprocessed result from C3.
C004 unprocessed result from C4.
C005 unprocessed result from C5.

P001 processed result from DEPTH(1) .
P002 processed result from DEPTH(2) .
P003 processed result from DEPTH(3) .
P004 processed result from DEPTH(4) .
P005 processed result from DEPTH(5) .

Suppose now that the programmer wishes to add another line of text to the print-out, saying "THE RESULTS ARE:". By using Line Insert, this text could follow line 3098. Alternatively, to illustrate the full effect of the PRINT "*string*" statement, the new text could be added as an extra line at the end of the program viz.

```
3098  PRINT "ALL TESTS PASSED"

3099  FOR I=1 TO 5

3100  PRINT DEPTH(I)

3101  NEXT

3102  PRINT "THE RESULTS ARE:"
```

The print-out will now be:

WATER LEVELS

ST3 4:54:36.2

* ALL TESTS PASSED

* THE RESULTS ARE: Note that the new text *precedes* the results but *follows* the previous text.

: 1st unprocessed result's text

C001 etc., results as before

PRINT *item* can include as many items as there is room for on the line, provided that they are separated by commas, i.e.:

PRINT *item,item,item . . . . . item.*

For example:

```
PRINT A,B,D, . . . . . Z
```

Note, however, that the print-out will have *all* the results numbered P001, i.e.

```
P001   result from A

P001   result from B
 :       :       :
 :       :       :
P001   result from Z
```

unless a Print Statement Extension is used, as described in Section 9.4.

This also applies to:

1010 PRINT A which gives P001 *result from* A

1011 PRINT B which gives P001 *result from* B

etc., unless "Extended".

[*device code*]
When setting-up the Task Definition, the user can select which output device is to receive the results. However, the Print Statement can include a code to direct the *processed* result to a different device. The original Channel result is still sent to the output device specified in the Task Definition. The code need only be included in the Statement if the output device required for the processed result is *different* to that specified for the associated Task.

The codes are as follows:

Action	Device Code
Print to Display	\ DI
Print to Printer	\ PR
Print to Cartridge	\CA
Print to GPIB	\ GP
Print to RS232	\ S0
Print to RS422/a	\ S1
Print to RS422/b	\ S2

For example, if the Task Definition directs all results to the RS232 port, but a Print Statement is included in the program as follows:

5015 PRINT \ DI ␣ MEAN

then the result from the variable MEAN will be sent *only* to the Display.

Only one code may be used in any one Print Statement, but results may be duplicated to several output devices by repeating the Statement for the result on as many lines of program as necessary, using a different device code in each.

[*modifier*]
When a modifier is included in a Print Statement, it notifies the processor that the output is to be treated as though it were in Alarm status. There are two modifiers, viz:

< which indicates low Alarm status
> which indicates high Alarm status

The Print Statement usually forms part of a conditional structure, e.g. an IF . . . THEN statement, in order to determine the Alarm status.

Example:

```
5050  IF LEVEL>6.4 THEN
5051  PRINT >␣ \PR␣LEVEL
```

In this example, all the other results could be sent to the cartridge, as declared in the Task Definition. However, if the result in LEVEL exceeds 6.4, the value will be sent to the *printer* in Alarm format, i.e. the print-out will be of the form:

```
!
P001 6.5      >.
```

Alarm text could be included by using a Print Statement Extension, as described in Section 9.4. If the result is from an *indexed* variable, the Alarm text could form part of the Output Format Declaration, as described in Section 9.5, in which case it would be called up automatically as a result of the Alarm status.

Note that when a line containing a modifier is being composed in the Display, a space is automatically inserted after PRINT, i.e. the Display will show:

```
PRINT␣ >.
```

If the line is subsequently recalled to the Display, the space will have been removed. There is no need to re-insert it.

Reverting to the general form of the Print Statement, i.e.:

PRINT␣ [*modifier*]␣ [*device code*]␣ *item, item, ... item*

Note that the modifier and the device code apply to *all* the items in the Statement.

Example Suppose that Channel 1 is measuring the voltage of a 5V supply, which should be providing a current of about 30mA, measured on Channel 2. The relevant Channel Definitions would state:

```
C1  VDC   Alarm High 5.1V
C2  mAd   Alarm High 35mA
```

There is an Alarm check made on each separate parameter, but not on their product, i.e. the power dissipation.

The processing program could contain a simple variable called POWER, whose value is defined on a particular program line as:

```
2020  POWER=C1*C2          followed by:
2021  PRINT POWER
```

Because no Output Format Declaration can be made for simple variables, line 2021 will merely cause the *value* of POWER to be printed, i.e. with no text, no units and no alarm facility.

However, suppose that two more simple variables, called LIMTLO for the low limit of POWER and LIMTHI for the high limit of POWER, are introduced into the program, e.g.

```
2020  POWER=C1*C2
2021  LIMTLO=125                    i.e. 125 mW
2022  LIMTHI=175                    i.e. 175 mW
2023  IF POWER>LIMTHI THEN
2024  PRINT >\GP POWER
2025  ELSE
2026  PRINT POWER
2027  ENDIF
```

This program offers several improvements over the previous version:

a. Line 2023 introduces a test for Alarm status by means of the IF . . . THEN Statement.

b. Line 2024, by including the ">", forces the output into the Alarm format as described in Chapter 11.

c. Line 2024, by means of the code \GP, directs the Alarm result to the GPIB.

d. Line 2026 ensures that if there is no Alarm, the value of POWER will be printed out on the device as defined by the Task.

This program can be improved even further by:

a. Extending the IF . . . THEN statement to include a test for LIMTLO.

b. Adding Extensions to the POWER print out, as described in Section 9.4, to make it more meaningful, e.g.:

```
2024 PRINT > \GP POWER;"DANGER";105;"MW"
```

which will cause the following line to be output to the GPIB if a high Alarm state is detected:

```
! DANGER P105 180 MW >.
```

9.4 PRINT STATEMENT EXTENSIONS

The previous Section describes the various forms of the Print Statement itself. However, with the exception of PRINT "*string*", the Print Statement can also be extended to contain the following items of information, which make the output results more meaningful.

a. Processed result number.
b. Units.
c. Processed result number and units.
d. Text and processed result number.
e. Text, processed result number and units.

For example, the simple program of Section 2 is reproduced below:

```
1000  A=C1*10

1001  PRINT A          ←Print Statement
```

This program produced a print-out of the processed result in the form:

```
ST1 12:13:36.6
:
P001  23.2
DT1
```

Note that this result is in Full format, as set by the Task Definition.

Changing the Processed Result Number

The processed result shown above bears the number P001. *All* processed results will bear this number, unless otherwise defined. To change the processed result number to, e.g., P002, extend line 1001 to:

```
1001   PRINT A;2
```

Note the use of the semi-colon as a separator. The extended Print Statement will change the previous print-out to:

```
ST1   12:13:36.6
:
P002   23.2
DT1
```

Adding Units

Units of up to three characters can be added to the processed result as follows:

```
1001   PRINT A;2;"DEG"
```

Note the additional semi-colon separator, and that the required units, in this case DEG, (for degrees), must be enclosed in inverted commas.

The print-out will now be:

```
ST1   12:13:36.6
:
P002   23.2 DEG
DT1
```

Adding Descriptive Text

Text, up to 18 characters, can be added to the print-out when in Full format:

```
1001   PRINT A;"WATER COOLANT";2;"DEG"
```

Note that the text must be enclosed in inverted commas.

This will produce the print-out:

ST1 12:13:36.6

: WATER COOLANT

P002 23.2 DEG

DT1

The text will *not* be printed if Compact format has been requested during Task Definition.

Extensions can be added to Print Statements anywhere in the program. Note that the three fields comprising the Extensions must always follow the Print Statement in the sequence:

Print Statement;"*Text*", *Processed Result Number*;"*Units*"

The previous section showed that a Print Statement can include several variables, for example:

4043 PRINT A,B,D,E, etc.

However, a Print Statement *Extension* applies only to the variable that immediately precedes it, e.g.:

4043 PRINT A,B,D;"TEXT";3;"UNITS",E

will produce:

P001 result from A

P001 result from B

P003 result from D with text and units

P001 result from E

Now that all the Print Statement options have been described, the general form of the Statement is repeated here:

PRINT [*modifier*] ␣ [*device*] ␣ *item;*[*extensions*],*item;*[*extensions*], . . . etc.

with an analysis of a complete example:

PRINT < ␣ \S0 ␣ A;"TEXT A";3;"UNITS A",B;"TEXT B";4;"UNITS B"

where:

a. The *modifier* < means: indicate a low Alarm state.
b. The *device* \S0 means: print to RS232.
c. The first *item* is the simple variable A.
d. The first *extensions* are "TEXT A";3;"UNITS A"
e. The second *item* is the simple variable B
f. The second *extensions* are "TEXT B";4;"UNITS B"

The Extensions are particularly useful with simple variables, which are not declared at the start of the program. They can be used in IF . . . THEN statements to enable Alarm text to accompany a processed result, particularly if the Print Statement includes a modifier, e.g.

```
2020  IF TEMP>100 THEN

2021  PRINT < TEMP;"WARNING";"DEG"
```

which would produce a print-out of the form:

```
!  WARNING

P001  104 DEG
```

Their application to indexed variables is explained in Section 9.7 where the Extensions are compared with Output Format Declarations.

9.5 OUTPUT FORMAT DECLARATION

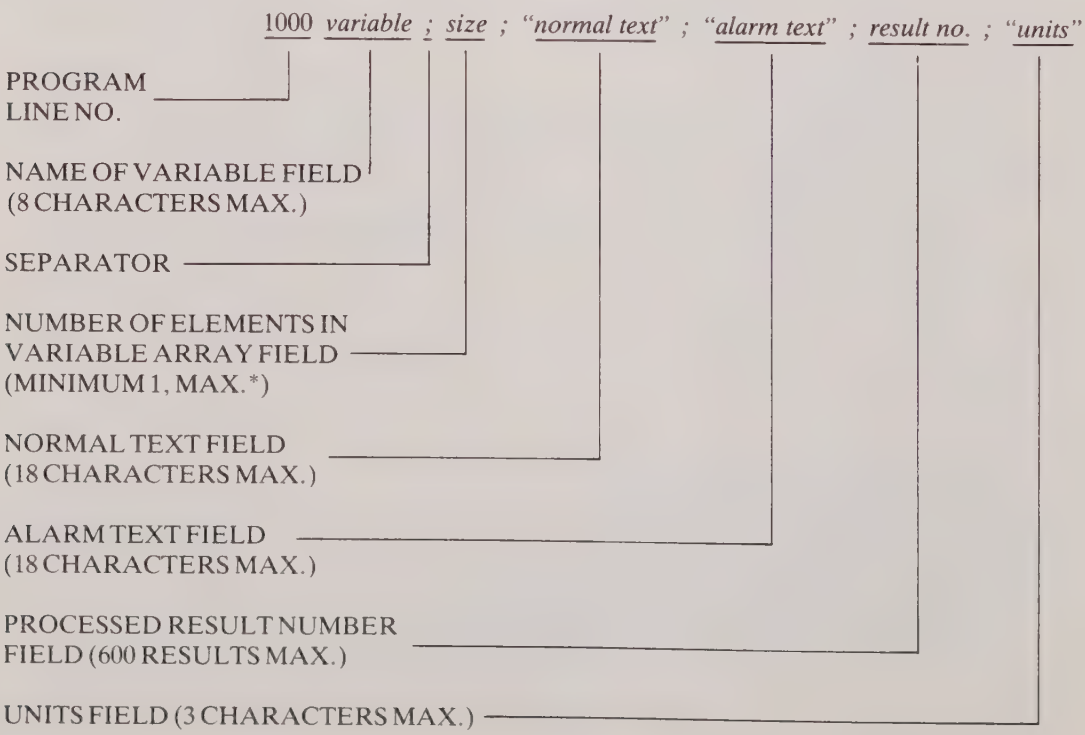
As explained in Section 6.2, *indexed* variables must be declared at the beginning of the program. This declaration is made in the general form:

Line number Variable;size

where *variable* represents the name of the indexed variable, e.g. MEAN, and *size* represents the number of elements in the array. The presence of the ";" separator is mandatory. A typical example is:

```
1000  MEAN;5
```

This declaration can be extended to contain much more information, in an Output Format Declaration. The items that can be included, and the sequence of fields is:

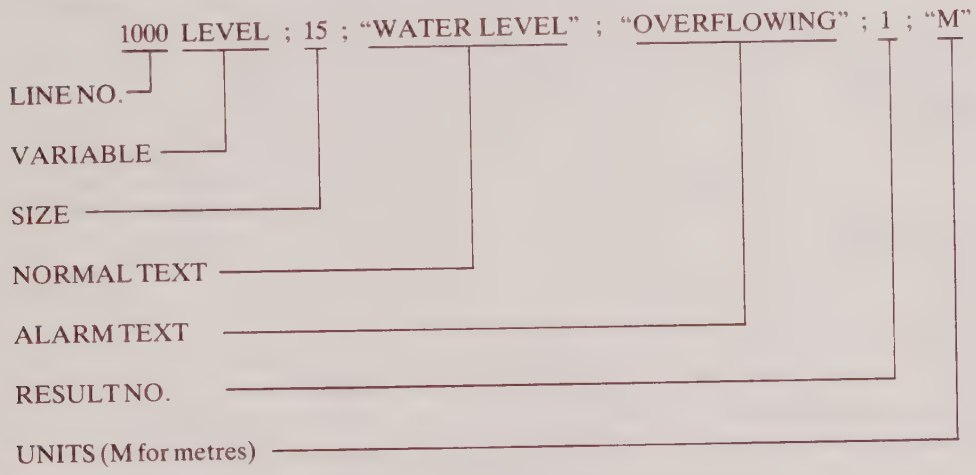

1000 variable ; size ; "normal text" ; "alarm text" ; result no. ; "units"
PROGRAM LINE NO.
NAME OF VARIABLE FIELD (8 CHARACTERS MAX.)
SEPARATOR
NUMBER OF ELEMENTS IN VARIABLE ARRAY FIELD (MINIMUM 1, MAX.*)
NORMAL TEXT FIELD (18 CHARACTERS MAX.)
ALARM TEXT FIELD (18 CHARACTERS MAX.)
PROCESSED RESULT NUMBER FIELD (600 RESULTS MAX.)
UNITS FIELD (3 CHARACTERS MAX.)

* The number of elements in an array is limited only by the amount of memory available in the logger. Memory availability reduces with the number of Options in the logger and the size of the program, see Section 15 "Memory Capacity".

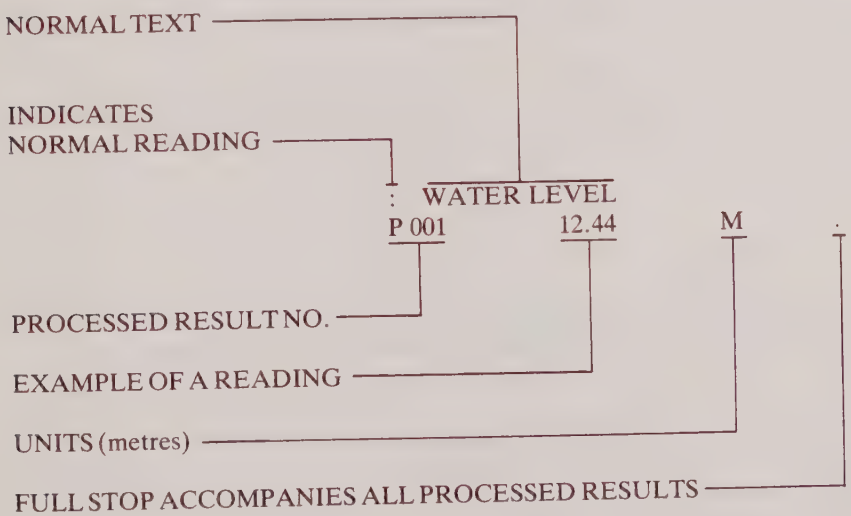
Note 1: Semi-colon separators must be included as shown.

Note 2: All text for printing must be in inverted commas.

Example 1 A program containing the variable LEVEL, with 15 elements, could start with the Declaration:



For a normal, i.e. non-alarm, reading, this will generate the following print-out for the first element in the array:



The equivalent alarm reading would be:

```
! OVERFLOWING
  P 001    15.00    M    .
```

The complete Declaration shown above can be reduced to the following combinations of items:

a. Processed result number.
b. Text.
c. Processed result number and units.
d. Text and processed result number.
e. Text, processed result number and units.

Text, in the above list, can be either "*normal*" or "*normal and alarm*".

However, if "*normal text*" is omitted, the inverted commas *must* still be included, as shown in example 2:

Example 2

1000 *variable; size; "";* *"alarm text";result no.;* *"units"*

Normal text omitted

Any text that is included in the declaration, e.g.:

LEVEL ; 3 ; "WATER LEVEL";1;"M"

will be printed out for *all three* LEVEL readings. However, if individual text is required for each reading, e.g. WATER LEVEL 1, WATER LEVEL 2, etc, this can be achieved by using the Print Statement Extensions, described in Section 9.4.

As also explained in Section 6.2, several indexed variable declarations may be included in the same line. However, the addition of full Output Format Declarations may restrict the programmer to one variable per line, because of the line length limitation.

Result numbering in Output Format Declarations is described in Section 9.6. The relative advantages of Output Format Declarations and Print Statement Extensions are described in Section 9.7.

9.6 RESULT NUMBERING

Output results can be numbered in one of the following two ways:

a. Results output from the pre-defined variables C1 to C600, whether processed or not, are *always* printed with the relevant variable number. For example, the result from C1 always appears in the form:

C001 21.43 RPM

followed by a full stop if the result has been processed. The user cannot influence this numbering, it is controlled entirely by the logger.

b. Results output from user-created variables always appear *after* the last of the results from the pre-defined variables, accompanied by a "P" number. For example:

P001 23.2 DEG

P numbers can be allocated within the range P001 to P999.

The P numbers *are* under the control of the programmer, and can be defined for *all* user-created variables by a Print Statement Extension, see Section 9.4. Also, for indexed variables only, they can be defined in an Output Format Declaration, as shown below.

Whichever method is used for defining them, once the first number has been declared for an indexed variable, the rest of the result numbers for that array will be allocated in sequence, e.g.:

```
POWER;5;101
```

will generate five lines of output, numbered P101 to P105 inclusive.

If no processed result number is declared, the processor will assign the default value of 1 to the first result, i.e. P001, and continue sequentially for subsequent results *in that array* with P002, P003 etc. However, if there are several variables declared at the start of the program, with undefined result numbers, e.g.:

```
1000  POWER;5
1001  LEVEL;5
1002  TEMP;5
etc.,
```

the five POWER results will be numbered P001 to P005, and the five LEVEL and five TEMP results will also be numbered P001 to P005. Therefore the user should write, e.g.:

```
1000  POWER;5;1     to number from P001 to P005
1001  LEVEL;5;6     to number from P006 to P010
1002  TEMP;5;11     to number from P011 to P015
etc.,
```

9.7 COMPARISON OF PRINT STATEMENT EXTENSIONS AND OUTPUT FORMAT DECLARATION

The Output Format Declaration is applicable *only* to indexed variables. However, to output results from an indexed variable, the program must also contain a Print Statement. Hence, the programmer who wishes to add text and units to these results, and also to influence the processed result numbers, has the choice of using the Output Format Declaration, a Print Statement Extension, or both, as follows:

a. Declare the full Output Format, i.e.:
variable;size;"normal text";"alarm text";result no;"units",
and augment the Print Statement to force Alarm status and select a different output device, if required.

b. Declare the full Output Format and use the Print Statement as for "a", but also use the Extensions to alter the *"text", result number and "units"* for specific processed results from each array element.

c. Simply declare the indexed variables *name* and *size* and use the Print Statement with Extensions to define the rest of the information for each array element.

The Print Statement Extensions take precedence over the Output Format Declaration. When the processor encounters a Print Statement involving an indexed variable in the program, it first looks to see if there are any Extensions to be actioned. If there are not, it looks to see if there is an Output Format Declaration. Failing both these, the processor outputs the results, with *no* text or units, numbering them from P001 upwards.

Note that when a Print Statement is used to output elements of an indexed variable, *any* Extension, such as a change of processed result number, automatically over-rides the *entire* Output Format Declaration for those elements.

The following examples of a 3-element indexed variable, called A, show the effects of the different methods:

1. The minimum declaration is:

   1000 A;3

2. The Statement PRINT A(I) will then output the three results:

   P001 result from A(I)

   P002 result from A(2)

   P003 result from A(3)

3. If line 1000 becomes:

   1000 A;3;101

   PRINT A(I) gives:

   P101 result from A(1)

   P102 result from A(2)

   P103 result from A(3)

4. With line 1000 as before, i.e.:

   1000 A;3;101

   but the Print Statement extended to:

   PRINT A(I);222

   this Extension for the result number over-rules line 1000 to give:

   P222 result from A(1)

   P222 result from A(2)

   P222 result from A(3)

This result highlights the best use of the combined Output Format Declaration and Print Statement Extensions, i.e.:

a. Use the Declaration to define *text, result numbers* and *units* for *all* the elements.

b. Reserve the Print Statement Extensions for single elements from the array. These can then be output with different *text*, *units* etc., to those in the Declaration, e.g.:

```
1000  A;3;"TEXT A";101;"UNITS"
```

with, later on, PRINT A(2);"TEXT *B*"102;"UNITS"

will give:

P101 result from A(1) with TEXT *A*

P102 result from A(2) with TEXT *B*

P103 result from A(3) with TEXT *A*

However, Print Statement Extensions can be configured to re-number results consecutively. For example, the Output Format Declaration could be:

```
1000  A;50;"TEXT A";101
```

followed by:

```
1040  FOR  I=1  TO  50
1041  PRINT  A(I)
1042  NEXT
```

This gives:

```
P101  result from A(1)   with TEXT A
  :          :                :
  :          :                :
P150  result from A(50)  with TEXT A
```

After further processing the programmer may wish to output a specially identified group from this array, say elements 10 to 19. This can be achieved as follows:

```
1080  FOR J=10 TO 19
1081  PRINT A(J);"TEXT B";J+x
1082  NEXT
```

where J+*x* is the processed result number, as follows:

Let *x*=200

This will give:

```
P210  result from A(10)  with TEXT B
  :          :                :
  :          :                :
P219  result from A(19)  with TEXT B
```

The expression J+*x* can include *any* operator.

Note that, as every indexed variable that is to be output must be declared and also have a Print Statement, both the methods discussed so far in this Section require a minimum of two lines of program, so neither appears to offer any saving. However, Section 9.5 shows that the Output Format Declaration can include *two* items of text, i.e. "*normal*" and "*alarm*", as opposed to only *one* item in each Print Statement Extension. This dual text feature can be used in conjunction with the "*modifier*" to the Print Statement, described in Section 9.3, as follows:

a. If no modifier is included in the Print Statement, and there is no text in the Extension, the result will be printed out with the "*normal*" text from the Declaration.

b. If an Alarm modifier is included in the Print Statement, and there is no text in the Extension, the result will be printed out with the "*alarm*" text from the Declaration.

For example, the Declaration could be:

```
1000  A;50;"ALL WELL";"DANGER";200;RPM
```

This would usually be followed later in the program by an IF . . . THEN statement, in order to determine the Alarm status, e.g.:

```
1030  IF A(3)>100 THEN

1031  PRINT>A(3)
```

If the result in A(3) exceeds 100, say 120, the print-out will contain:

```
!  DANGER

P203  120 RPM
```

9.8 WRITE STATEMENTS

The WRITE statement enables *text messages* to be sent to the output device specified in the Task Definition. A device code may be included to send the message to a different output device.

The main differences between the Print and Write Statements are that the latter:

a. Are treated as text only, although this text may include a calculation, and are not in ORION result format.

b. Do *not* appear within the relevant Start and Done Task markers. They are output as soon as possible after the processor has encountered them in the program. Their precise position in the sequence of output data is difficult to predict, owing to the multi-tasking nature of the system software, but it is likely that the message will appear *before* the standard scan output.

As a simplified example to compare the effect of the Print and Write Statements, assume that C1 contains the value 5 and that the printer has been declared in the Task Definition. The line:

```
3030  PRINT C1*10
```

will produce a print-out of the form:

```
ST1

P001   50

DT1
```

However, the line:

```
3030   WRITE  C1*10, "IS THE ANSWER"
```

will produce the result:

```
50      IS THE ANSWER
```

somewhere on the print-out.

The general form of the Write Statement is:

WRITE[*device code*] ␣ *item,item,item,* . . . . etc.,

as explained below.

[*device code*]
The device code defines the output device to which the message is to be sent, as follows:

Action	Device Code
Write to Display	\ DI
Write to Printer	\ PR
Write to Cartridge	\ CA
Write to GPIB	\ GP
Write to RS232	\ S0
Write to RS422/a	\ S1
Write to RS422/b	\ S2

The code must be included in the statement only if the output device required for the message is different to that specified for the associated Task. For example, if the Task Definition directs all results to the RS232 port, and the WRITE statement message includes the code for GPIB, then the message will be sent to the GPIB *only*.

Only one code may be used in any one WRITE statement. Messages may be duplicated to several output devices, however, by repeating the WRITE statement for the message on as many lines of program as necessary, using a different device code in each. Note that these messages cannot be sent to the cartridge of ORION D in Coded format, and subsequently decoded, due to their text only content.

Items
The item to be written may be one of two types:

a. An *expression*, which, by the definition in Section 3.4, may be any variable or a mathematical or Boolean expression which evaluates to a numeric result.

As shown as part of an earlier example in this Section, the programmer can include the line:

```
3030 WRITE C1*10
```

where C1 contains the value 5. This will simply cause the value 50 to appear somewhere in the output, with no text, units, or result number. This form of Write Statement therefore has fairly limited applications. Note that as C1*10 is not in inverted commas, the processor knows that it is to Write the *value* contained in C1 multiplied by ten, rather than treat the statement C1*10 as text.

b. A "*string*" which may be any string of characters enclosed in quotation marks.

A practical application of the Write Statement, using both the above types would be to include it in an IF . . . THEN statement.

e.g.
```
2020 IF C21>5 THEN
2021 WRITE \ DI "WARNING AT", TIMER0, "SECONDS"
2022 ENDIF
```

Assume that all other information from Task 2 is being sent to the RS232 port. Line 2021 will cause the numeric value accumulated in Timer 0 to be added to the text and the whole message to be sent to the Display, where it will appear in the form:

```
WARNING AT 26.0000 SECONDS
WARNING AT 28.0000 SECONDS
```

etc., updated according to the scanning rate of Task 2.

When the Write Statement includes several items, as shown in the general form earlier in this Section, a comma must be included between each pair as a separator. This comma also causes each item to be added to the output message at specific "tab" positions in the standard 80-character output format, as shown below:

```
00..... 11..... 22..... 33..... 44..... 55..... 66..... 77..
tab     tab     tab     tab     tab     tab     tab     tab
```

Each number, when printed, is limited to the maximum numeric field size of ten characters, plus an automatically inserted space. Each value of a succession of values in the same message will begin on a tab mark, e.g.:

368 . . . . . 4840 . . . . . 50 . . . . . etc.

A *string* occupies its precise length, e.g. the message "RUN AWAY" occupies eight character positions. The next message, in the same Statement, would start on tab mark 11. This message could be "TAKE ␣ REMEDIAL ␣ ACTION", which occupies 20 characters. A third message in the Statement would therefore begin on tab mark 33. Any message longer than 3 characters commencing on tab mark 77 will be truncated.

Unlike PRINT, the word WRITE is not available on a single key and must therefore be written letter by letter from the keypad. The whole work must be preceded and followed by a space.

9.9 KILL

The KILL statement can be used to delete Channel results from the stream of data which is fed to the output device. This facility is useful where "All" data has been selected for logging in the Task but certain results are not required to be output. For example, the user may be interested only in the *processed* results of certain Channels, and hence the original readings may be discarded.

A KILL'd results storage space is made available for a fresh result *immediately* the statement is executed. This helps to maintain the scanning rate when the measurement buffer, described in Chapter 1 Section 3.2, is approaching an overload condition.

The general form of the statement is:

KILL *p,q,r,s,* . . . etc.,

the number of Channels included in each KILL statement being limited only by the line length capacity of the Display.

The statement is applicable solely to the pre-defined variables C1 to C600. Only the element number must be quoted, e.g. to remove the results of Channels 1, 2 and 3 write:

KILL 1,2,3 *not* KILL C1,C2,C3

Where consecutive Channels are to be KILL'd, e.g. Channels 7, 8, 9, 10 and 11, an indexed form can be used in a FOR . . . NEXT loop:

```
6066  FOR I=7 TO 11
6067  KILL (I)
6068  NEXT
```

Where the Channels are in a non-consecutive sequence, e.g. 5, 13, 19, 48, 92, with no other intervening numbers, the whole sequence can be KILL'd by using the Scanned Channel form SN(I), as described in Section 13.

Reverting to a simple example given previously, where the logger measures current on Channel 1 and voltage on Channel 2. The user is interested only in the product of these, i.e. the power. Therefore it was suggested that the programmer could write:

```
C1=C1*C2
```

thus re-using the pre-defined variable C1 to store the power value, and, at the same time, deleting the redundant current value. This still leaves the unwanted voltage value in C2, but by now writing:

```
KILL 2
```

the storage space for the original Channel 2 result is immediately made available for re-use.

9.10 SET . . . TO

The SET . . . TO statement can be used to influence the analogue and digital outputs of the logger as follows:

Analogue Outputs

SET *analogue channel* TO *value*. This instruction sets the output of the specified Analogue Channel to the value defined. Chapter 8, "Using the Analogue Output Module", explains how the output of each Analogue Channel can be set by defining the value required as a percentage of the maximum obtainable.

The SET---TO instruction can either set the output to a fixed value, or cause it to follow the values of one of the program variables, as shown in the examples below. These examples assume that Channels 61 to 65 are Analogue Outputs, set up as current sources.

Example 1 Fixed Value Output. The instruction:

```
1008  SET 63 TO 50
```

causes Channel 63 to produce 50% of its output. As a current source, the output can be varied between 4 and 20mA, i.e. a change of 16mA. 50% of this is 8mA, which, when added to the minimum of 4mA equals 12mA.

Example 2 Output Controlled by Program Variable

```
1008  SET 63 TO SETPOINT(3)
```

Every time the program is executed, the output of Channel 63 will be set to the value currently stored in the variable SETPOINT(3). Note that this value must also be a percentage, which must have been calculated earlier in the program prior to being stored in SETPOINT(3).

This facility can be extended by using a FOR---NEXT loop:

```
1008  FOR I=1 TO 5
1009  SET (I+60) TO SETPOINT(I)
1010  NEXT
```

Here, Channels 61 to 65 inclusive will have their outputs set to the values stored in the variables SETPOINT(1) to SETPOINT(5) respectively.

Digital Outputs

The state of any output of the Digital Input-Output Module can be changed by a program statement of the form:

SET *digital channel* **TO** *value*

where:

digital channel represents the number of the required Digital Output Channel, and:

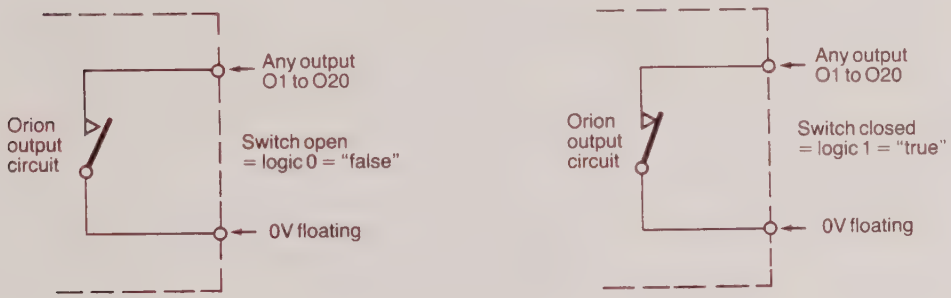
value represents an expression which the processor will resolve into either Logic 1 or Logic 0, as described later.

More than one Channel can be SET, as shown under SET . . . TO *string*.

Logic 0 and Logic 1

The outputs of the Digital I/O Module are at Logic 0 in the quiescent state. The user can choose the actual voltage level of logic 0, as described in Chapter 6. The alternative state to logic 0 is Logic 1, which is also known as the "TRUE" state. Logic 0 is then referred to as the "FALSE" state.

A simple way to remember the convention for the two logic states is to regard each output as being controlled by a switch. The conditions for one such switch are shown below:



Logic Output States

Suppose that Channels 41 to 60 are Digital I/O type. An example of a simple digital output statement is:

```
3004  SET 41 TO 1
```

This statement sets the output of Channel 41 to Logic 1, i.e. the switch is closed. Conversely:

```
3022  SET 41 TO 0
```

sets the output of Channel 41 to Logic 0, i.e. the switch is open.

Note that any outputs which have been SET TO 1 automatically revert to Logic 0, i.e. the switches open, when the logger is halted.

The only numeric value which is *unique* is zero, represented by Logic 0. *Every* value other than zero is interpreted by the processor as being Logic 1, e.g.:

```
3004  SET 41 TO 6-1     or:
3004  SET 41 TO 36
```

are both interpreted as:

```
SET  41 TO 1
```

As another example of the use of processing applied to Digital Outputs, a Channel can be used to indicate equality between two analogue inputs, viz:

```
3004  SET 41 TO ABS(C1-C2)<1E-4
```

This line is interpreted by the processor as follows:

1. Calculate the difference between the values of Channels 1 and 2.
2. Take the *absolute* value of this difference, i.e. treat all results as positive.
3. To avoid problems of resolution to several decimal places, compare the absolute value with 1E-4, i.e. 0.0001.
4. If the absolute value is less than 1E-4, the inputs must be approximately equal. Also, the expression:
   ```
   ABS(C1-C2)<1E-4
   ```
   is TRUE, and the processor resolves it to read:
   ```
   SET 41 TO 1
   ```
   which closes the switch.
5. Conversely, if the absolute value is greater than 1E-4, the inputs are not equal. Also, the expression is FALSE, and the processor resolves it to read:
   ```
   SET 41 TO 0
   ```
   which opens the switch.

SET *digital channel* **TO** *"string"*. Several consecutive Digital Channels can have their values included in a "string" instruction. For example:

```
3004  SET 41 TO "10110"
```

causes the following output pattern to be set up:

Channel	41	42	43	44	45
Value	1	0	1	1	0

Notes:
1. Only the first Channel in the sequence must be specified in the instruction.
2. The inverted commas around the string are essential, SET 41 to 10110 is interpreted by the logger as SET 41 TO ten thousand one hundred and ten, which being non-zero, sets the output of Channel 41 to Logic 1.
3. Only constants may be included in the string.

Alternative Form. An alternative form of instruction for consecutive channels is:

```
3004  SET 41 TO 0,1,0,1 (the commas are essential).
```

which produces the output pattern:

Channel	41	42	43	44
Value	0	1	0	1

This form may include calculations, e.g.

```
3004  SET 41 TO C1<C2,C3<C4,C5<C6,C7<C8.
```

If C1 is less than C2, the expression C1<C2 is TRUE, but if C1 is greater than or equal to C2, the expression is FALSE. A TRUE result produces a Logic 1, a FALSE result produces a Logic 0. The same rules apply to the other three comparisons, and the outputs will be set accordingly.

Indexing in a FOR---NEXT loop may also be employed, viz:

```
3004  FOR I=1 TO 10
3005  SET (40+I) TO SC(I)>LIMIT(I)
3006  NEXT
```

10 DIGITAL INPUTS

Processing can be performed on digital signals, either as single bit inputs or as groups.

Single Bit Inputs
Each single bit input is regarded as being either a Logic 1 or a Logic 0. The two Boolean operators AND and OR can be used in the IF . . . THEN statements described in Section 8.1.

For example:

```
3005  IF C21=1 THEN....
```

or:

```
3005  IF C21=1 AND C22=1 THEN....
```

or:

```
3005  IF C21=1 OR C22=0 THEN....
```

The next example of single bit input processing includes an application of the dual use of the "=" sign, as discussed in Section 3.5 "Assignments". Consider the following statement, where C21 and C22 are Status inputs, monitoring switch closures in the external plant.

```
3008  RESULT=C21=C22
```

Assignment

Relational,
by virtue of its position *following* the assignment.

As described in Chapter 5:

a. When the plant switch is closed, a logic 0 state is detected.
b. When the plant switch is open, a logic 1 state is detected.

Hence, if both switches are closed, C21 and C22 are both at logic 0, whereas if both switches are open, C21 and C22 are both at Logic 1. In either case the equation is TRUE, C21 equals C22, which yields a logic 1 as the result. Both cases therefore simplify the original line to:

```
RESULT=1
```

i.e. the processor assigns the value "logic 1" to the variable RESULT.

Conversely, when one switch is open and the other closed, C21 cannot equal C22. The equation is FALSE and a logic 0 is produced as the result. The original line now reduces to:

```
RESULT=0
```

and the processor assigns the value "logic 0" to the variable RESULT.

Another example is:

```
CONDMET=C21 AND C22 AND C23=0
```

As before, the equation can be either TRUE or FALSE, and hence CONDMET will be assigned either a 1 or a 0.

This line of program would probably be followed by, e.g.,

```
IF CONDMET=1 THEN      etc.
```

Grouped Inputs
Digital inputs can be grouped together in the Status/Event Channel Definition shown in Chapter 9. For example, suppose that the inputs of Channels 21 to 24 are to be grouped for processing. Channel 1 only must be declared in the Channel Definition. When the Sensor menu offers "Select digital input function", press the soft key "bit", and enter the number "4".

The required input condition could be:

C24	C23	C22	C21
1	1	0	1

The grouping process causes these inputs to be regarded as a binary number, in this case:

8	4	2	1
1	1	0	1

which equals 13 (i.e. 8+4+1)

The programmer can now write:

```
IF C21=13 THEN      proceed accordingly.
```

This method is much more compact than the alternative of testing for four single bits, which would be written:

```
IF C21=1 AND C22=0 AND C23=1 AND C24=1 THEN      etc.
```

11 CONTROL OF OTHER TASKS

It is possible to control the activity of other Tasks by means of flexible cross-triggering and cross-aborting statements, which may be included within IF . . . THEN control structures.

11.1 TRIGGERING OTHER TASKS

TRIGGER *x,y,z* . . . etc. This statement will cause Tasks *x*, *y* and *z* to be triggered. The trigger will simply be ignored if the Task is already active. Up to all 8 Tasks may be specified in the Task list, although self-triggering is of little practical use.

11.2 ABORTING OTHER TASKS

ABORT *p,q,r* . . . etc. This statement will cause Tasks *p*, *q*, and *r* to be aborted, i.e. stopped. Up to all 8 Tasks may be included in the Task list, and self-aborting is acceptable.

NB Due to internal software time delays and the Task prioritisation, it is possible for statements of this type to depart from their apparent sequence. For example, Task A may issue a statement to TRIGGER Task B chronologically *after* a statement to ABORT Task B has been issued by Task C. However, owing to the aforementioned delays etc., the TRIGGER does not happen as it is almost instantly *followed* by the ABORT statement.

12 COUNTERS AND TIMERS

ORION *delta* contains eight Counters and nine Timers, as described in the following two sections. Although they can be regarded as indexed variables, the processor treats them as special cases in that the inclusion of parentheses round individual element numbers is optional, e.g. COUNT4 and COUNT(4) are equally valid. However, the general indexed form must still include parentheses, i.e. COUNT(I) and TIMER(I). Counter and Timer variables are all global.

12.1 COUNTERS

Each Task has its own Counter which totals the number of scans performed in that Task since RUN was operated. The Counters are automatically reset to start from zero each time that RUN is commanded. Note that counters cannot be written to, i.e. they cannot be preset to an initial count. The maximum count is 2^{31} i.e. 2,147,483,648, which would be displayed as 2.147E9.

The functions available are:

COUNT1 to COUNT8 inclusive, which are Scan Counters for Tasks 1 to 8.

COUNT(I) which affords indexed access to Counters 1 to 8 inclusive.

To determine the total held in any of the Counters since RUN was operated, e.g. in Counter No. 7, simply write:

```
5008  PRINT  COUNT7
```

An example of the use of a Counter in the calculation of a running average of Channels 1 to 10 in Task 6, over several scans, is given below:

```
6007  FOR  I=1  T0  10

6008  SUM(I)=SUM(I)+C(I)

6009  AVERAGE(I)=SUM(I)/COUNT6

6010  PRINT  AVERAGE(I)

6011  NEXT
```

Note that COUNTS are updated at the beginning of a scan. If the interval between scans is small, a second scan may have commenced before the first has been fully processed, as shown in Fig. 12.3. In this case, the COUNT reading indicates the number of scans started but not necessarily completed.

For ease of identification, the sign of the COUNT reading goes −ve when the number of scans requested in the Task Definition is reached. This also occurs if the Task is aborted. However, if the Task is aborted and then re-triggered, the sign will revert to positive, and the count will continue from the total reached prior to the abort command.

12.2 TIMERS

The functions available are:

TIMER0 which automatically records time elapsed since RUN was operated, for the whole logger. This function is resettable.

TIMER1 to TIMER8 inclusive. These Timers can be allocated to *any* Task, all eight to the same Task if so desired.

TIMER(I) affords indexed access to Timers 0 to 8 inclusive.

The Timers count in seconds, with a resolution of 1/10th of a second, to a maximum of 2^{31}, which represents approximately 6.8 years! If a print-out of a long time period is required, the processing facility can be used to convert the result into hours, minutes and seconds. Normally, however, hours and minutes information would be taken from the Start Task and Done Task Markers.

The nine Timers operate as follows:

TIMER0 is reset to zero when RUN is pressed, and also by executing any Statement which assigns a value to it, the most logical being TIMER0=0. The Timer would also be reset by e.g. TIMER0=42, but note that the value 42 is *not* preset into it. As soon as the Timer has been reset it automatically starts again from zero.

TIMERS 1 to 8 are started by executing any statement which assigns a value to them, they are *not* started by RUN or by the start-up of any Task which uses them. To start e.g. TIMER5 write TIMER5=0. As with TIMER0, if TIMER5=42 is written, TIMER5 will be reset to *zero*, not preset to 42.

All the Timers can be interrogated as often as required e.g.:

```
7009  IF TIMER0>10.5 THEN
7010  PRINT "TIME EXCEEDED"
7011  ENDIF
```

A print-out of a Timer can be obtained by writing e.g. PRINT TIMER4

but for use in calculations within the logger, it is advisable to 'freeze' the time at a particular instant by writing e.g. TIME=TIMER4, then use the value in TIME for the calculation.

13 RANDOM CHANNEL ACCESS

Hitherto in this Chapter, Channels used in program examples have been referred to either as individual named elements, e.g. C1, C8 etc., or as an indexed variable in the form C(I). This form of indexed Channel variable is most useful where:

a. The Channels are scanned consecutively. In this case, when used in a FOR . . . NEXT loop, STEP is assumed to be equal to +1.

b. The Channels are scanned non-consecutively, but with constant spacing between them, e.g. 1, 3, 5, 7, 9. In this example, a FOR . . . NEXT loop would have to include the declaration STEP 2, so that every alternate Channel is selected.

The use of STEP etc., is explained in Section 8.2 "FOR . . . NEXT Loops".

However, the Channels to be processed may have been scanned in an irregular, non-consecutive sequence such as:

8,14,30,64-69, etc.

Although the individual Channels are still accessible as C8, C14, etc., the C(I) form of indexed variable becomes unwieldy. Therefore, for sequences of this type, the following indexed variables gives access to the Ith Channel in the set of scanned Channels.

SC(I) Ith Channel result in scan set

SH(I) High Limit for Ith Channel in scan set

SL(I) Low Limit for Ith Channel in scan set

SN(I) Gives Channel number of Ith Channel scanned.

A simple application of the Scanned Channel facility would be to print the results from the previously mentioned sequence: 8,14,30,64-69, i.e. 9 Channels in all. The program would be:

```
1000  FOR I=1 TO 9
1001  PRINT SC(I)
1002  NEXT
```

Note that I = 1 TO 9 *not* 8 TO 69

The FOR . . . NEXT loop behaves as described in Section 8.2, except that in line 1001, when I=1, it is not Channel 1 that is printed but the *first* Channel in the scan set, i.e. Channel 8. Similarly, when I=2, the *second* Channel, 14, is printed etc. Putting it in general terms, it is the *I*th Channel in the scan that is accessed.

All other FOR . . . NEXT loop facilities still apply, e.g. it is still permissible to write: FOR I=2 TO 4, in which case only the 2nd, 3rd and 4th Channels will be accessed. Also, STEP can still be used, as in:

```
FOR I=1 TO 9 STEP 2
```

in which case the 1st, 3rd, 5th, 7th and 9th Channels will be accessed.

Example: Assume that Task 1 scans the five non-consecutive Channels 5,9,11,17,19. A program to record the cumulative total value of each Channel, over several scans, is given below.

```
1000  TOTAL;5
1001  FOR I=1 TO 5
1002  TOTAL(I)=TOTAL(I)+SC(I)
1003  PRINT TOTAL(I)
1004  NEXT
```

With the exception of the SC(I) function, the main features of the program have been described in earlier sections. However, for completeness, they are summarised again in the following overall description.

Immediately RUN is pressed to start scanning, the declaration in line 1000 is read. This line instructs the processor to allocate storage space for an indexed variable called TOTAL, which consists of five elements forming the variables TOTAL(1) to TOTAL(5). These five variables all have an initial value of zero, and they remain in existence until RUN is operated again to stop scanning.

The rest of the program is executed once for each time that Task 1 causes the Channels to be scanned, but line 1000 is not actioned again during these repetitions.

The remainder of the program is a FOR . . . NEXT loop, which behaves as described in sub-section 8.2, but contains a novel feature in line 1002. Remembering that SC(I) refers to the Ith Channel rather than to the Channel number itself, line 1002 is effectively expanded by the processor to mean:

TOTAL(1)=TOTAL(1)+1ST SCANNED CHANNEL

TOTAL(2)=TOTAL(2)+2ND SCANNED CHANNEL

TOTAL(3)=TOTAL(3)+3RD SCANNED CHANNEL

TOTAL(4)=TOTAL(4)+4TH SCANNED CHANNEL

TOTAL(5)=TOTAL(5)+5TH SCANNED CHANNEL

Reading the first line as an example, from left to right, it means:

"the value to be stored in TOTAL(1) is to be the value already in there plus the current value of the first scanned Channel, i.e. Channel 5".

On the first scan, the value already in TOTAL(1) is zero, so the new value is simply that produced by the first measurement on Channel 5. Similarly TOTAL(2) will contain the first value measured on Channel 9, and so on for the other three Channels.

After the second scan, the five variables will contain the sum of the values measured on their respective Channels over the first two scans. The summation continues for as many scans as the Task commands.

Line 1003 causes the values of the five variables to be printed out after every scan. These results can have additional information included, either by using the Print Statement Extensions or the Output Format Declaration. The latter could be:

1000 TOTAL;5;"TOTAL FLOW";"HIGH FLOW";1;"LTR"

where LTR represents litres.

The print-out of the *n*th scan, with "All" data logged, would then resemble:

```
                    ST1     12:45:56.3
:                   C005    7.145     LTR
:                   C009    1.223     LTR
:                   C011    22.85     LTR
:                   C017    52.50     LTR
:                   C019    1.104     LTR
: TOTAL FLOW        P001    301.5     LTR  .
: TOTAL FLOW        P002    532.0     LTR  .
: TOTAL FLOW        P003    612.3     LTR  .
: TOTAL FLOW        P004    32.85     LTR  .
: TOTAL FLOW        P005    111.6     LTR  .

                    D.T.1
```

Note that although all five processed results are accompanied by the common, non-alarm text "TOTAL FLOW", each one has an individual result number P001 to P005, which identifies the first to the fifth scanned Channel respectively.

Because the SC(I) function refers to the Ith Channel scanned, rather than to the specific Channel number, the same program can be used for any five Channels simply by altering the Task Definition to scan the new Channels. However, to aid in identifying the different sets of readings on the print-out, the processed result numbers could be changed. For example, the second set of five readings could be numbered from P006 to P010 by changing the Output Format Declaration to:

```
1000  TOTAL;5;"TOTAL FLOW";"HIGH FLOW;006;"LTR"
```

although the processed result numbers begin to lose their association with the Channels from which they were derived.

The SC(I) function also allows access to a mixture of 3-pole and 6-pole Channels in a scan set.

SN(I)

SN(I), which stands for Scanned channel Number (I), can be used to ascertain the numbers of the scanned Channels. For example, if the Channels scanned are 1, 4, 7, 16 etc., the program line:

```
PRINT \ DI SN(3)
```

will cause the number of the third Channel, i.e. 7, to appear in the Display when the Task is run. The Print Statement is described in Section 9.3.

Another application of the SN(I) function is to enable a non-consecutive sequence of Channel results to deleted from the output queue, using the KILL statement described in Section 9.9. For example, to KILL results from Channels 5, 13, 19, 48, 92, i.e. 5 all told, write:

```
1015  FOR I=1 TO 5
1016  KILL SN(I)
1017  NEXT
```

As the index numbers 1 to 5 are progressively substituted in line 1016, the meaning of the statement becomes: KILL the result from the 1st Channel in the scan, i.e. 5; KILL the result from the 2nd Channel, i.e. 13 etc. The processor actually simplifies this to: KILL 5, KILL 13, KILL 19 etc., which is the correct form for the KILL statement.

Note, however, that if SC(I) were used instead of SN(I), then, when I=1, line 1016 would be interpreted as:

> KILL the result from the Channel whose number is contained as the value stored from the 1st Channel scanned. This Channel, 5, may have produced a result such as 1048.6, in which case the line becomes:
>
> > KILL the result from Channel 1048.6!

14 STORING AND REPLAYING PROCESSED DATA AND PROGRAMS

The use of the integral cartridge recorder of ORION D is described in Chapter 11 Section 3. As explained there, a maximum of seven "programs" can be recorded on one cartridge. A program, in this sense, consists of the Channel and Task Definitions, plus any processing programs that are included in the Tasks.

When recording logged data, whether processed or not, the Full and Compact format records are in normal ASCII and hence the cartridge capacity is as given in Chapter 11, Section 3, "Recording Format". The Coded format, however, is in a special form of ASCII which enables up to 50,000 records to be accommodated on one cartridge. When *processed* results are recorded in Coded format, the following conditions apply:

1. If the results are in the pre-defined variables C1 to C600, each result constitutes only one record on the cartridge.

2. If the results are the *derived* processed type, from user-created variables, e.g. A, TEMP, DEPTH(I) etc., then each one is automatically accompanied by a "pointer" record to assist in decoding. This has two implications:

   a. The cartridge can now accommodate fewer results.

   b. When Coded processed data is being replayed to the logger from the cartridge, the original processing program *must* be present in the memory, to enable the pointers to be decoded.

15 MEMORY CAPACITY

The amount of free memory available to the user reduces with the number of Options in the logger, and the size of any program already entered. Memory availability can be calculated from the following data:

1. With the maximum of 600 Channels fitted, there are 8,976 bytes of free memory available.
2. Each of the slots 4 to 13 inclusive that is not occupied releases a further 520 bytes of memory, but the slots must be depopulated from the top, i.e. the highest number. Removing intermediate Channel cards does *not* release memory.
3. Each line of program, however short, consumes 88 bytes.
4. Each simple variable of name length L characters consumes L + 6 bytes.
   e.g. The simple variable MEAN consists of 4 characters, so will consume 4 + 6 i.e., 10 bytes.
5. Each indexed variable of name length L characters and consisting of N elements consumes
   $L + 12 + (N \times 4)$ bytes
   e.g. The indexed variable PRESSURE (6) has a name length of eight characters and consists of six elements. It will therefore consume $8 + 12 + (6 \times 4)$ i.e., 44 bytes.

16 VOCABULARY OF FUNCTIONS

ORION Delta can recognise the words and symbols shown in the following alphabetical list. Each term is fully explained in the section indicated.

TERM	SECTION
ABORT	11.2
ABS(x)	7.2
AND	7.3
ATAN(x)	7.2
C1 to C600	2.4, 6.3
C(I)	6.3
Constants of the following types:	
Exponent form	5,
Messages	5,
Numeric	5,
Text	5, 9
Units	5, 9
COS(x)	7.2
COUNT1 to COUNT8	12.1
COUNT(I)	12.1
ELSE	8.1
ENDIF	8.1
EXP(x)	7.2
FOR	8.2
GOTO	8.3
H1 to H600	6.3
H(I)	6.3
IF	8.1
Indexed Variable Name, up to 8 characters	6.6
INT(x)	7.2
KILL	9.9
L1 to L600	6.3
L(I)	6.3
LN(x)	7.2
Messages	5, 9.3, 9.8

TERM	SECTION
NEXT	8.2
Operators of the following types:	
Boolean	7.3
Integer Exponentiation	7.1
Mathematical	7.1
Relational	7.3
OR	7.3
Parentheses ( )	7.1
PRINT	9.3
SC(I)	13
SET	9.10
SH(I)	13
SIN(x)	7.2
SL(I)	13
SN(I)	13
SQR(x)	7.2
STEP	8.2
Text	5, 9
THEN	8.1
TIMER0	12.2
TIMER1 to TIMER8	12.2
TIMER(I)	12.2
TO	8.2, 9.10
TRIGGER	11.1
Units	5, 9
Variable Name, up to 8 characters	6.6
WRITE	9.8

17 EXAMPLE PROGRAMS

17.1 CONTENTS

The following example programs are given in this Section:

1. Single Channel Average over 5 Readings
2. Average of 3 Channels on Every Scan, with a Cumulative Average of Each Channel Every Five Scans
3. Maximum and Minimum Reading
4. Output on Change (as shown in Chapter 10, Fig. 10.4)
5. Output on Change, Giving Old and New Values and the Difference Between Them
6. Warning if Channel Reading is Outside Limits
7. Alarm Threshold (as shown in Chapter 10, Figs. 10.2 and 10.3)
8. Walking Window Average on One Channel
9. Walking Window Average on Three Channels
10. Mean and Standard Deviation

17.2 INTRODUCTION

For simplicity, most of the programs are shown using a minimum number of Channels. However, once the basic principles have been understood, the programs can be extended to encompass many more Channels and to include additional statements. As with any software language, there are probably several ways to achieve the desired result; the examples given are suggestions only.

As explained in Section 2, every Channel whose result is to be processed must be "configured" in the Channel Definition. If this is not done, the Channel listing will appear as:

```
CHANNEL  001
measurement UNDEFINE
D
conversion 0
alarm Off
o/p 000,000
analogue o/p  0
text 0
alarm text  0
```

All the examples in this Section use a minimum Channel Definition of the form:

```
----------------------
CHANNEL  001
measurement Volts
 DC
conversion 0
alarm Off
o/p 000,000
analogue o/p  0
text 0
alarm text  0
```

Each program is accompanied by a typical print-out, and also the Task Definition which was set up to produce it. This Definition may be amended, if so desired, e.g. to include different Channels, perform more scans, output all results rather than just processed ones etc.

The programs are written, for clarity, on a one statement per line basis. However, in practice, they could be considerably condensed by using multi-statement lines.

17.3 SINGLE CHANNEL AVERAGE OVER 5 READINGS

TASK DEFINITION

```
TASK  001
title TIME AVERAGE 1

Function Scan
trigger Timer
delay 00-00:00:00.0
scans 5
scan Inter
interval 00-00:00:01
.0
channels 001
rate 10/s
processing On
data to be logged Pr
oc
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
1000 SUM=SUM+C1
1001 COUNTER=COUNTER
+1
1002 IF COUNTER=5 TH
EN
1003 AVERAGE=SUM/5
1004 PRINT AVERAGE:"
TIME AVERAGE";1;"VDC
"
1005 COUNTER=0
1006 SUM=0
1007 ENDIF
```

PRINT-OUT

```
RUN   16:44:19 06-03
  TIME AVERAGE 1
S T 1 16:44:23.7
 :TIME AVERAGE
P 001 4.23186  VDC .

D T 1

HALT  16:44:34 06-03
```

PROGRAM KEY

1 All variables start with value zero when RUN is pressed to start scan.

2 On the first scan, SUM stores the value measured on Channel 1. On the second scan, SUM retains the previous value, and adds to it the new value from Channel 1. This process continues for all the 5 scans specified in the Task Definition. SUM then holds the sum of all five readings on Channel 1.

3 On the first scan, COUNTER stores the value "1". On the second scan, COUNTER stores the value 1 + 1, i.e. 2.
This process continues for all 5 scans, until COUNTER holds the value 5.

4 On the first 4 scans, COUNTER cannot contain 5, therefore the processor ignores all intervening lines and jumps straight to the ENDIF in line 1007. On the 5th scan, COUNTER will contain 5, so the rest of the program will be actioned.

5 The total value stored in SUM is divided by 5, and the result, i.e. the required average value, is stored in the new variable AVERAGE.

6 On the 5th scan only, the value in AVERAGE is output, in this case to the Printer, as specified in the Task Definition.

7 The contents of the two variables is set to zero. Strictly speaking, there is no need to do this, as the Task is limited to 5 scans. To obtain the average value from a further 5 scans, RUN would have to be pressed again to halt the logger, and again to re-start, so the variables would automatically have zero value. However, it is good practice to clear variables, once they have served their purpose. Note that any reading stored in AVERAGE will be cleared automatically, whenever a new value is assigned to it.

8 Title from Task Definition.

9 Title from line 1004 of program.

10 Processed result, which is the required average reading from Channel 1 over 5 scans.

17.4 AVERAGE OF 3 CHANNELS ON EVERY SCAN, WITH A CUMULATIVE AVERAGE OF EACH CHANNEL EVERY FIVE SCANS

TASK DEFINITION

```
TASK  001
title GROUP/TIME AVE
RAGE
Function Scan
trigger Timer
delay 00-00:00:00.0
scans *
scan Inter
interval 00-00:00:02
.0
channels 005,011,018

rate 10/s
processing On
data to be logged Pr
oc
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
1000 TIMESUM(3)
1001 GSUM=0
1002 FOR I=1 TO 3
1003 GSUM=GSUM+SC(I)

1004 TIMESUM(I)=TIME
SUM(I)+SC(I)
1005 NEXT
1006 PRINT GSUM/3;"G
ROUP AVERAGE";1;"VDC
"

1007 TALLY=TALLY+1
1008 TOTAL=TOTAL+1
1009 IF TALLY=5 THEN

1010 FOR I=1 TO 3
1011 AVER=TIMESUM(I)
/TOTAL
1012 PRINT AVER;"TIM
E AVERAGE";SN(I);"VD
C"
1013 NEXT
1014 TALLY=0
1015 ENDIF
```

PRINT OUT

Over 5 scans. Note that first 4 scans are Group Average only; the 5th scan also gives the cumulative averages.

```
RUN    01:42:39 01-01
  GROUP/TIME AVERAGE
S T 1 01:42:39.9
 :GROUP AVERAGE
P 001 4.87366  VDC .

D T 1
  GROUP/TIME AVERAGE
S T 1 01:42:41.9
 :GROUP AVERAGE
P 001 4.87366  VDC .

D T 1
  GROUP/TIME AVERAGE
S T 1 01:42:43.9
 :GROUP AVERAGE
P 001 4.87366  VDC .

D T 1
  GROUP TIME AVERAGE
S T 1 01:42:45.9
 :GROUP AVERAGE
P 001 4.87366  VDC .

D T 1
  GROUP/TIME AVERAGE
S T 1 01:42:47.9
 :GROUP AVERAGE
P 001 4.87366  VDC .
 :TIME AVERAGE
P 005 6.43100  VDC .
 :TIME AVERAGE
P 011 2.22630  VDC .
 :TIME AVERAGE
P 018 5.52400  VDC .

D T 1
```

PROGRAM KEY

1 Note non-consecutive Channel numbers.

2 A 3-element variable, to store results from the Channels for cumulative averages.

3 This variable is assigned the value 0, to zero it for each scan prior to receiving the Group Sum results.

4 After each scan, GSUM holds the sum of the readings from the 3 Channels, i.e. the Group Sum. Note the use of SC(I) to read the non-consecutive Channels 5,11 and 18.

5 TIMESUM(1),(2) and (3) store the readings from the 1st, 2nd and 3rd scanned Channels respectively.

6 Output the Group Sum divided by 3, i.e. the Group Average, to the device selected in the Task Definition, in this case the Printer.

7 Records the groups of 5 scans.

8 Records the total number of scans.

9 Start of an IF . . . THEN loop to output the average reading from each Channel on every 5th scan. The loop is *not* actioned on the first 4 scans; the program skipping straight to ENDIF on line 1015.

10 Note that this is the second time that I has been used as an index in the program. This is good economy, as it minimises the number of variables to be stored. However, I can only be used again here because there is no conflict with the previous application.

11 The FOR . . . NEXT loop is executed 3 times. On the first execution, the sum of the readings stored in TIMESUM(1) is divided by the number of scans stored in TOTAL. The result is stored in the simple variable AVER (for "average"), thus leaving the accumulated sum of the readings undisturbed in TIMESUM(1). The Print Statement causes the value of AVER to be added to the queue of data awaiting outputting, thus releasing AVER ready to accept the average value from TIMESUM(2)/TOTAL on the second execution of the loop. A similar sequence occurs for TIMESUM(3).

Remember that PRINT expression must *never* start with an indexed variable, i.e. the temptation to simply write:

PRINT TIMESUM(I)/TOTAL must be firmly resisted!

The use of SN(I) to define processed result number enables the original Channel identity to be preserved, as shown in the results print-out.

17.5 MAXIMUM AND MINIMUM READINGS, ON ONE CHANNEL, OVER 10 SCANS

TASK DEFINITION

```
TASK  001
title MAX/MIN

Function Scan
trigger Timer
delay 00-00:00:00.0
scans 100
scan Inter
interval 00-00:30:02
.0
channels 001
rate 10/s
processing On
data to be logged Pr
oc
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
1  1000 IF N=0 THEN
   1001 MAX= -100000
   1002 MIN=100000
   1003 ENDIF
2  1004 N=N+1
3  1005 IF C1>MAX THEN
   MAX=C1
   1006 ENDIF
4  1007 IF C1<MIN THEN
   MIN=C1
   1008 ENDIF
   1009 IF N=10 THEN
   1010 PRINT MIN;"MIN
5  VALUE";1;"VDC"
   1011 PRINT MAX;"MAX
   VALUE";2;"VDC"
   1012 N=0
   1013 ENDIF
```

PRINT-OUT

Over 30 scans.

```
RUN    16:30:41 07-03
  MAX/MIN
S T 1 16:30:59.4
 :MIN VALUE
P 001 0.0384200VDC .
 :MAX VALUE
P 002 0.153280 VDC .

D T 1
RUN    16:33:22 07-03
  MAX/MIN
S T 1 16:33:40.6
 :MIN VALUE
P 001 0.096510 VDC .
 :MAX VALUE
P 002 0.264130 VDC .

D T 1
RUN    16:35:24 07-03
  MAX/MIN
S T 1 16:35:42.9
 :MIN VALUE
P 001 0.759320 VDC .
 :MAX VALUE
P 002 1.52168  VDC .

D T 1
```

PROGRAM KEY

1 This IF . . . THEN statement is executed only on the first scan after RUN has been pressed. The effect is to pre-set the variable MAX to a very small value, and MIN to a very large value. This gives the processor two initial values against which to compare the first readings.

2 The variable N acts as a tally counter, incrementing by 1 on every scan.

3 Whenever the reading on Channel 1 is greater than that already stored in MAX, the latter is replaced by the new reading.

4 Whenever the reading on Channel 1 is less than that already stored in MIN, the latter is replaced by the new reading.

5 This IF . . . THEN statement is actioned only on the 10th scan. The minimum and maximum values, stored in MIN and MAX respectively, are output to the device specified in the Task Definition, in this case the Printer.

N is reset to zero, ready for the next 10 scans.

17.6 OUTPUT ON CHANGE, FOR ONE CHANNEL

TASK DEFINITION

```
TASK  004
title CHANGE

Function Scan
trigger Timer
delay 00-00:00:00.0
scans *
scan Inter
interval 00-00:00:02
.0
channels 001
rate 10/s
processing On
data to be logged Pr
oc
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PRINT-OUT

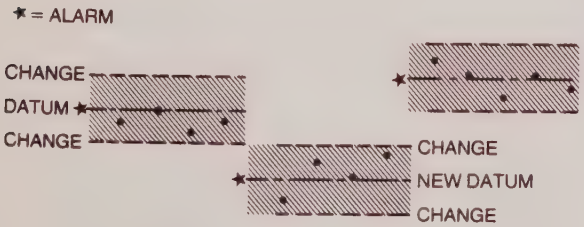
```
4000 IF FLAG=0 THEN
4001 FLAG=1
4002 DATUM=C1
4003 CHANGE=0.01
4004 PRINT C1;1;"VDC
"
4005 ENDIF
4006 IF C1>DATUM+CHA
NGE OR C1<DATUM-CHAN
GE THEN
4007 DATUM=C1
4008 PRINT C1;1;"VDC
"
4009 ENDIF
```

PROGRAM

```
RUN    11:29:20 08-03
  CHANGE
S T 4 11:29:20.9
 :
P 001 1.33100  VDC .

D T 4
  CHANGE
S T 4 11:31:09.7
 :
P 001 1.34200  VDC .

D T 4
  CHANGE
S T 4 11:32:27.3
 :
P 001 1.40500  VDC .

D T 4
  CHANGE
S T 4 11:33:42.4
 :
P 001 1.39400  VDC .

D T 4
  CHANGE
S T 4 11:34:34.3
 :
P 001 0.998000 VDC .

D T 4
```

"Change" is defined in Chapter 10, Section 3, the relevant part being reproduced here for convenience:

Change (Fig. 10.4) A value of change is entered. The first result always constitutes an alarm, and the value of this result becomes the datum level for observing changes. Thereafter, an alarm occurs only when the result lies outside the datum ± change levels. When an alarm occurs, the datum moves to the value of the alarm result, and subsequent readings are compared with the new datum ± change. Each subsequent alarm causes the datum to adopt the new alarm value.



SUCCESSIVE RESULTS FROM ONE CHANNEL

PROGRAM KEY

1 This IF . . . THEN statement is actioned only on the first scan after RUN has been pressed, i.e. this is the only time that FLAG=0.
DATUM is set to the first reading on Channel 1.
The minimum change that will be recognised is defined as 0.01; the units being Volts in this case, as declared in the Channel Definition.
The initial value of C1 is output to the device specified in the Task Definition, in this case the Printer.

2 If a reading on C1 differs from the DATUM value by more than the permitted CHANGE amount, this reading becomes the new DATUM value, and is also output to the Printer.

17.7 OUTPUT ON CHANGE, GIVING OLD AND NEW VALUES, AND THE DIFFERENCE BETWEEN THEM

TASK DEFINITION

```
TASK  001
title CHANGE

Function Scan
trigger Timer
delay 00-00:00:00.0
scans *
scan Inter
interval 00-00:00:02
.0
channels 001
rate 10/s
processing On
data to be logged Pr
oc
header LogData
output device Print
format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
      1000 IF FLAG=0 THEN
      1001 FLAG=1
1     1002 DATUM=C1
      1003 CHANGE=0.01
      1004 PRINT DATUM;"NE
      WDATUM";2;"VDC"
2     1005 ELSE
3     1006 DIFF=ABS(C1-DAT
      UM)
4     1007 IF DIFF>CHANGE
      THEN
5     1008 OLDATUM=DATUM
6     1009 DATUM=C1
      1010 PRINT OLDATUM;"
      OLDATUM";1;"VDC"
7     1011 PRINT DATUM;"NE
      WDATUM";2;"VDC"
      1012 PRINT DIFF;"DIF
      FERENCE IS";3;"VDC"
      1013 ENDIF
      1014 ENDIF
```

PRINT-OUT

```
RUN    17:10:16 08-03
  CHANGE
S T 1 17:10:16.8
 :NEWDATUM
P 002 5.00100  VDC .

D T 1
  CHANGE
S T 1 17:14:16.0
 :OLDATUM
P 001 5.00100  VDC .
 :NEWDATUM
P 002 5.01200  VDC .
 :DIFFERENCE IS
P 003 0.0109996VDC .

D T 1
  CHANGE
S T 1 17:15:08.6
 :OLDATUM
P 001 5.01200  VDC .
 :NEWDATUM
P 002 4.98300  VDC .
 :DIFFERENCE IS
P 003 0.0290007VDC .

D T 1
  CHANGE
S T 1 17:15:48.2
 :OLDATUM
P 001 4.98300  VDC .
 :NEWDATUM
P 002 5.00400  VDC .
 :DIFFERENCE IS
P 003 0.0210008VDC .

D T 1
  CHANGE
S T 1 17:16:31.1
 :OLDATUM
P 001 5.00400  VDC .
 :NEWDATUM
P 002 5.01600  VDC .
 :DIFFERENCE IS
P 003 0.0119998VDC .

D T 1
```

PROGRAM KEY

See previous program for description of purpose.

1 As for previous program.

2 Program jumps straight to this line on all scans other than the first.

3 The difference (DIFF) is defined as the *absolute* value of the difference between the latest reading on Channel 1 and the previously stored reading in DATUM. Note that polarity is disregarded.

4 Only if the difference is greater than the permitted change, i.e. 0.01, will the rest of the program be executed. If the difference is less than CHANGE, the program is terminated for that scan, and nothing is printed.

5 The previous value in DATUM is transferred to OLDATUM.

6 The new reading on Channel 1 is stored in DATUM.

7 The three results, i.e. the old reading, the new reading and the difference between them, are output to the device specified in the Task Definition, in this case the Printer.

17.8 WARNING IF CHANNEL READING IS OUTSIDE LIMITS

TASK DEFINITION

```
TASK  001
title WARNINGS

Function Scan
trigger Timer
delay 00-00:00:00.0
scans *
scan Inter
interval 00-00:00:02
.0
channels 001
rate 10/s
processing On
data to be logged Pr
oc
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
1 1000 HIGH=5.2
2 1001 LOW=4.7
3 1002 IF C1>HIGH THEN

4 1003 PRINT> C1;"WARN
  ING HIGH LIMIT";1;"V
  DC"
  1004 ELSE
5 1005 IF C1<LOW THEN

6 1006 PRINT< C1;"WARN
  ING LOW LIMIT";1;"VD
  C"
  1007 ENDIF
  1008 ENDIF
```

PRINT-OUT

```
RUN   09:38:30 08-03
  WARNINGS
S T 1 09:38:30.9
 !WARNING HIGH LIMIT
P 001 5.21032  VDC>.

D T 1
  WARNINGS
S T 1 09:38:01.8
 !WARNING LOW LIMIT
P 001 4.69999  VDC<.

D T 1
```

PROGRAM KEY

1 High limit is defined.

2 Low limit is defined.

3 If the Channel reading is greater than the high limit:

4 Output the result, in alarm format.

5 If the Channel reading is less than the low limit:

6 Output the result, in alarm format.

17.9 ALARM THRESHOLD

TASK DEFINITION

```
TASK  001
title THRESHOLD

Function Scan
trigger Timer
delay 00-00:00:00.0
scans *
scan Inter
interval 00-00:00:02
.0
channels 001
rate 10/s
processing On
data to be logged Pr
oc
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

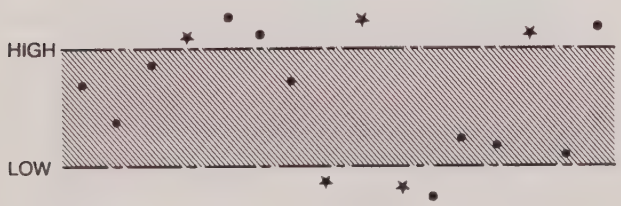
```
  1000 IF START=0 THEN
1
  1001 START=1
  1002 HIGH=5.2
  1003 LOW=4.8
  1004 ENDIF
  1005 IF C1>HIGH AND
2 FLAG<>1 THEN
  1006 PRINT> C1;">HIG
  H THRESHOLD";1;"VDC"

3 1007 FLAG=1
  1008 ENDIF
  1009 IF C1<LOW AND F
4 LAG<>2 THEN
  1010 PRINT< C1;"<LOW
   THRESHOLD";1;"VDC"
5 1011 FLAG=2
  1012 ENDIF
```

PRINT-OUT

```
RUN   14:06:58 08-03
  THRESHOLD
S T 1 14:06:58.8
 !>HIGH THRESHOLD
P 001 5.21000  VDC>.

D T 1
  THRESHOLD
S T 1 14:08:10.2
 !<LOW THRESHOLD
P 001 4.79000  VDC<.

D T 1
  THRESHOLD
S T 1 14:09:21.2
 !>HIGH THRESHOLD
P 001 5.22300  VDC>.

D T 1
  THRESHOLD
S T 1 14:10:03.7
 !<LOW THRESHOLD
P 001 4.88900  VDC<.

D T 1
```

PROGRAM KEY

This program is based on a combination of the Alarm thresholds illustrated in Chapter 10 Section 3, figs. 10.2 and 10.3. A high limit and a low limit are defined, the area between them constituting a "dead band". The first reading outside this band is output, but thereafter a reading will be output only when the entire band has been crossed, as shown below:



* = ALARM

SUCCESSIVE RESULTS FROM ONE CHANNEL

1 Actioned solely on the first scan, the only time when START=0. The high and low limits are defined.

2 If the reading on Channel 1 exceeds the high limit *and* FLAG does not equal 1, output the result in alarm format. Note that on the first scan, FLAG will be zero, therefore one of the conditions has been met, and the decision whether to output or not will depend solely on the reading.

3 If a high alarm has been registered, FLAG is assigned the value 1. This ensures that no further high alarms can be output until FLAG has been assigned a value other than 1, either by a low alarm or by stopping and re-starting scanning again via the RUN key.

4 If the reading on Channel 1 is less than the low limit *and* FLAG does not equal 2, output the result in alarm format. Note that on the first scan, FLAG will be zero, therefore the conditions are similar to those for line 1005 described above. However, the reading cannot be simultaneously *above* the high limit and *below* the low limit, so only one of the IF . . . THEN statements will be actioned.

5 If a low alarm has been registered, FLAG is assigned the value 2. This ensures that no further low alarms can be output until FLAG has been assigned a value other than 2, either by a high alarm or by stopping and re-starting scanning again via the RUN key.

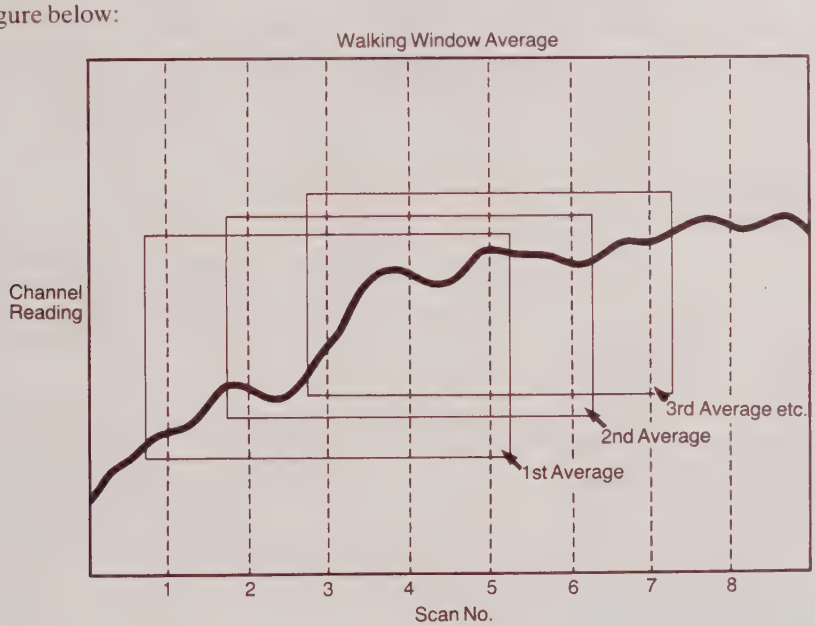
17.10 WALKING WINDOW AVERAGES

Previous examples show two methods of calculating averages:

1. Calculating a running average, with an output on every scan. When used for detecting trends, this method has the disadvantage that, once a large number of readings has been accumulated, each new reading has a barely perceptible effect on the result. Any trend becomes apparent only after several readings of either increasing or decreasing value have been added.

2. Calculating the average over groups of, say, 5 scans, with an output on every fifth scan. This method highlights trends much more clearly than the previous one, but has the disadvantage that a delay of 5 scans must elapse between consecutive readings. This could allow a trend to develop to dangerous extent before it is detected.

The walking window average offers a solution to these problems. An arbitrary number of readings is selected, say 5. Assume for the moment that the first 5 scans have been completed, the average calculated and the result output. On the 6th scan, the reading from scan 1 is discarded and the average of the readings from scans 2 to 6 is calculated, as shown in the figure below:



This process is repeated so that only the average of the latest 5 scans is output after each scan.

During the first scan, a special section of the program is used to ensure that there are always *five* readings to be averaged, even though only the first scan has been performed. In this case, the first reading is stored in the memory five times, to simulate five consecutive readings. The average is then calculated from these, thus yielding the result of just the first measurement.

After the second scan, the average is calculated from one scan 2 reading and four scan 1 readings. The table below shows how the "average" from the first 5 scans is calculated.

AFTER SCAN NUMBER:	NUMBER OF READINGS FROM SCAN:				
	1	2	3	4	5
1	5	–	–	–	–
2	4	1	–	–	–
3	3	1	1	–	–
4	2	1	1	1	–
5	1	1	1	1	1

17.11 WALKING WINDOW AVERAGE ON ONE CHANNEL

TASK DEFINITION

```
TASK  001
title WALKING WINDOW

Function Scan
trigger Timer
delay 00-00:00:00.0
scans *
scan Inter
interval 00-00:00:02
.0
channels 001
rate 10/s
processing On
data to be logged Al
l
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
1  1000 WINDOW;5
   1001 IF POINTER=0 TH
2  EN
   1002 FOR I=1 TO 5
   1003 WINDOW(I)=SC1
   1004 NEXT
   1005 ENDIF
3  1006 POINTER=POINTER
   +1
   1007 IF POINTER>5 TH
4  EN
   1008 POINTER=1
   1009 ENDIF
5  1010 WINDOW(POINTER)
   =SC1
   1011 AVERAGE=0
   1012 FOR I=1 TO 5
6  1013 AVERAGE=AVERAGE
   +WINDOW(I)
   1014 NEXT
7  1015 AVERAGE=AVERAGE
   /5
8  1016 PRINT AVERAGE;"
   AVERAGE";1;"VDC"
```

PRINT-OUT

C = Channel Reading
P = Processed Average Reading

```
RUN    14:50:35 13-03
  WALKING WINDOW
S T 1 14:50:35.7
 :
C 001 1.30007  Vdc
 :AVERAGE
P 001 1.30007  VDC .

D T 1
  WALKING WINDOW
S T 1 14:50:37.7
 :
C 001 1.40011  Vdc
 :AVERAGE
P 001 1.32008  VDC .

D T 1
  WALKING WINDOW
S T 1 14:50:39.7
 :
C 001 1.50025  Vdc
 :AVERAGE
P 001 1.36011  VDC .

D T 1
  WALKING WINDOW
S T 1 14:50:41.7
 :
C 001 1.40010  Vdc
 :AVERAGE
P 001 1.38012  VDC .

D T 1
  WALKING WINDOW
S T 1 14:50:43.7
 :
C 001 1.30007  Vdc
 :AVERAGE
P 001 1.38012  VDC .

D T 1
  WALKING WINDOW
S T 1 14:50:45.7
 :
C 001 1.20002  Vdc
 :AVERAGE
P 001 1.36011  VDC .

D T 1
  WALKING WINDOW
S T 1 14:50:47.7
 :
C 001 1.09989  Vdc
 :AVERAGE
P 001 1.30007  VDC .

D T 1

HALT   14:51:02 13-03
```

PROGRAM KEY

1 Declaration of 5-element variable, WINDOW(1) to WINDOW(5).

2 Used on first scan only, as described in previous section, to ensure that all elements contain a reading.

3 "Points out" which element is to store the reading.

4 Resets the Pointer after every fifth scan.

5 As the index "POINTER" is incremented from 1 to 5, so WINDOW becomes WINDOW(1), WINDOW(2) etc., and successive scans cause values of Scanned Channel 1 to be stored in the correct "WINDOW".

6 At the end of each scan, the new sum of WINDOWS(1) to (5) will have been stored in AVERAGE.

7 AVERAGE now contains the average value of the last 5 readings.

8 Outputs the average value to the device specified in the Task Definition, in this case the Printer.

Note that the Task Definition specifies that "All" data is to be logged, hence each scan produces a print-out from the Scanned Channel, in this case Channel 1, and a processed result showing the average value.

17.12 WALKING WINDOW AVERAGE ON THREE CHANNELS, OVER FIVE READINGS

TASK DEFINITION

```
TASK  001
title WINDOW (3-CHAN
NEL)
Function Scan
trigger Timer
delay 00-00:00:00.0
scans *
scan Inter
interval 00-00:00:02
.0
channels 003,007,012

rate 10/s
processing On
data to be logged Al
l
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
1  1000 WINDOW:15
  /1001 IF POINTER=0 TH
  |EN
  |1002 FOR N=1 TO 3
  |1003 FOR I=1 TO 5
2 <1004 WINDOW((N-1)*5+
  |I)=SC(N)
  |1005 NEXT
  |1006 NEXT
  \1007 ENDIF
3  1008 POINTER=POINTER
   +1
  /1009 IF POINTER>5 TH
4 <EN
  \1010 POINTER=1
   1011 ENDIF
   1012 FOR N=1 TO 3
   1013 SUM=0
5  1014 WINDOW((N-1)*5+
   POINTER)=SC(N)
  /1015 FOR I=1 TO 5
6'<1016 SUM=SUM+WINDOW(
  |(N-1)*5+I)
  \1017 NEXT
7  1018 SC(N)=SUM/5
   1019 NEXT
```

PRINT OUT

```
RUN    16:33:59 13-03
   WINDOW (3-CHANNEL)
S  T 1  16:33:59.9
  :
C  002 0.310451 Vdc .
  :
C  006 0.323107 Vdc .
  :
C  014 0.317229 Vdc .

D  T 1
   WINDOW (3-CHANNEL)
S  T 1  16:34:01.9
  :
C  002 0.306507 Vdc .
  :
C  006 0.316905 Vdc .
  :
C  014 0.312652 Vdc .

D  T 1
   WINDOW (3-CHANNEL)
S  T 1  16:34:03.9
  :
C  002 0.307233 Vdc .
  :
C  006 0.315985 Vdc .
  :
C  014 0.313516 Vdc .

D  T 1

ETC
```

This program is similar to the previous one, but shows how to create a "two-dimensional" array so that readings from more than one Channel can be logged. The variable, WINDOW, has 15 elements, arranged in 3 rows of 5 each, as shown in the figure:

Scanned Channel	WINDOW				
First (e.g. C2)	1	2	3	4	5
Second (e.g. C6)	6	7	8	9	10
Third (e.g. C14)	11	12	13	14	15

PROGRAM KEY

1 Declaration of 15-element variable, WINDOW(1) to WINDOW(15).

2 Used on first scan only. The two nested FOR . . . NEXT loops ensure that all 15 elements contain a reading, for reasons explained in Section 17.10. Line 1004 behaves as follows:

When N=1, (N−1)*5=0.
Therefore, as I is incremented from 1 to 5, the reading from Scanned Channel 1 is stored in WINDOW(1) to WINDOW(5).

When N=2, (N−1)*5=5. Therefore, as I is incremented from 1 to 5, the reading from Scanned Channel 2 is stored in WINDOW(6) to WINDOW(10); and so on.

3 Points to the relevant column in the array, on each scan.

4 Resets POINTER after every fifth scan.

5 This line behaves in a similar manner to line 1004, except that POINTER is now used as part of the index for WINDOW. For example, on the third scan of a group of five, POINTER will equal 3. Hence, when N=1, WINDOW(3) will store the reading from Scanned Channel 1. When N=2, WINDOW(8) will store the reading from Scanned Channel 2, and when N=3, WINDOW(13) will store the reading from Scanned Channel 3.

6 This FOR . . . NEXT loop is nested within another loop which starts on line 1012. When the outer loop sets N to, say 2; the inner loop will cause the values in WINDOW(6) to WINDOW(10) to be added together and stored in SUM.

7 When n=2, for example, the latest reading stored from Scanned Channel 2, which is now no longer required, is replaced by the average value of the last 5 readings on Scanned Channel 2.

Note that no Print Statement is used; the three averages stored in SC(1), (2) and (3) are output as defined by the Task Definition. Hence, they appear as "C" results, although accompanied by the dot which denotes a processed result.

17.13 MEAN READING AND STANDARD DEVIATION OF 20 CHANNELS, OVER 10 SCANS

TASK DEFINITION

```
TASK  001
title STD.DEV.AND ME
AN
Function Scan
trigger Timer
delay 00-00:00:00.0
scans 10
scan Inter
interval 00-00:00:02
.0
channels 001-020
rate 40/s
processing On
data to be logged Pr
oc
header LogData
output device Print
Format Full
tasks triggered 0
tasks aborted 0
```

PROGRAM

```
1  1000 MEAN(20)"MEAN"(
   1)"VDC"
2  1001 DEV(20)"SD"(1
3  1002 N=N+1
   1003 FOR I=1 TO 20
4  1004 MEAN(I)=MEAN(I)
   +C(I)
5  1005 DEV(I)=DEV(I)+C
   (I)**2
   1006 NEXT
6  1007 IF N=10 THEN
7  1008 FOR I=1 TO 20
8  1009 DEV(I)=MEAN(I)*
   MEAN(I)/N-DEV(I)
   1010 DEV(I)=SQR(ABS(
   DEV(I)/N))
9  1011 MEAN(I)=MEAN(I)
   /N
10 1012 PRINT MEAN(I),D
   EV(I)
11 1013 MEAN(I)=0
   1014 DEV(I)=0
   1015 NEXT
12 1016 N=0
   1017 ENDIF
```

PRINT OUT

FIRST 10 CHANNELS

```
RUN    00:39:24 01-A1
  STD.DEV.AND MEAN
S T 1 00:39:42.1
 :MEAN
P 001 1.65300  VDC .
 :SD
P 001 1.069E-3     .
 :MEAN
P 002 2.84200  VDC .
 :SD
P 002 8.734E-4     .
 :MEAN
P 003 0.918000 VDC .
 :SD
P 003 3.088E-4     .
 :MEAN
P 004 1.04400  VDC .
 :SD
P 004 5.348E-4     .
 :MEAN
P 005 3.55100  VDC .
 :SD
P 005 0            .
 :MEAN
P 006 2.21300  VDC .
 :SD
P 006 1.069E-3     .
 :MEAN
P 007 0.774000 VDC .
 :SD
P 007 0            .
 :MEAN
P 008 1.14400  VDC .
 :SD
P 008 6.176E-4     .
 :MEAN
P 009 1.97300  VDC .
 :SD
P 009 0            .
 :MEAN
P 010 2.06700  VDC .
 :SD
P 010 1.381E-3     .
```

PROGRAM KEY

1 Declares a variable, MEAN, having 20 elements (one per Channel).

2 Declares a variable, DEV, also having 20 elements, with normal text SD (i.e. Standard Deviation).

3 Tally Counter.

4 The readings from C1 to C20 are stored as variables MEAN(1) to MEAN(20).

5 The *squares* of the readings from C1 to C20 are stored as variables DEV(1) to DEV(20).

6 For the first 9 scans, N does not equal 10 and therefore the program jumps to line 1017 and terminates. On the 10th scan, the results are to be output and therefore the rest of the program is actioned.

7 Start of a FOR . . . NEXT loop, to process the stored results from all 20 Channels.

8 These two lines constitute an algorithm which re-arranges the text-book equation for Standard Deviation:

$$s = \sqrt{\sum_{i=1}^{n} \frac{(x_i - \bar{x})^2}{n}}$$

into a machine-compatible form.

Analysis of line 1009 reveals that DEV(I) will contain:

$$\frac{(\Sigma x_I)^2 - \Sigma x_I^2}{N}$$

and line 1010 modifies this to give the square root of the absolute value, which, when divided by N, represents the Standard Deviation.

9 The mean value of the readings from each Channel is stored in MEAN(1) to MEAN(20).

10 Print Statement.

11 Set variables to zero after print-out.

12 Set tally counter to zero after 10th scan.

Chapter 13 Specification

MEASUREMENT SPEED

Nominal (display menu)	**500/s**	**100/s**	**40/s**	**10/s**
Typical,* solid-state	500/s†	130/s	40/s	10/s
Typical,* reed relay	140/s	80/s	35/s	10/s
Integration time	0.625ms	5ms	20ms	80ms
Nominal scale	1000	10 000	100 000	100 000

†450/s for channels in slave scanners.
*This is the speed of uninterrupted scanning, measuring channels and putting the measurements into memory.

Typical system throughput
100 channels, 20 continuous scans at 500/s nominal.
Output binary encoded to cartridge, RS422 or GPIB.

	Solid-state		**Reed relay**	
Nominal scan speed	500/s	100/s	500/s	100/s
Maximum throughput	150/s	100/s	110/s	70/s
With limits, all chans output	130/s	100/s	110/s	70/s
Th-couples, no limits, all output	100/s	90/s	90/s	65/s
Th-couples, with limits, all output	90/s	85/s	85/s	65/s

When 40/s or 10/s are selected the throughput is essentially the same as the typical scan speed.

DC VOLTAGE

Nominal Speed		**500/s**	**100/s**	**40/s**	**10/s**
Range	Nominal Full Scale	Sensitivity			
100V§	110V	100mV	10mV	1mV	1mV
10V	13.7V	10mV	1mV	100μV	100μV
1V	1.7V	1mV	100μV	10μV	10μV
100mV	200mV	100μV	10μV	1μV	1μV
10mV	25mV	10μV	1μV	1μV†	1μV†

§ Not available with solid-state selectors.
† Scale length 10 000 nominal, but the measurement includes a 6th digit (100nV) to preserve the best resolution in processing results.

Limits of Error: for 1 year, 20 ± 3°C, 10/s.
10mV to 10V ranges: < ± (0.005% rdg + 2 digits + 1μV)
Temp. coeff: < (0.001% rdg + 0.1μV)/°C
100V range: < ± (0.05% rdg + 2 digits)
Temp. coeff: < 0.003% rdg /°C

DC CURRENT: data for 100Ω shunt.
Using connector 35303B, which includes 100Ω shunts, quoted errors apply for currents < 40mA. Connector 35303A may be used with external shunts.
Reading Rates and Scale Length as above.

Ranges:	100mA	10mA	1mA	100μA
Nominal full scale:	137mA	17mA	2mA	250μA
Sensitivity at 10 and 40/s:	1μA	100nA	10nA	10nA†

† Scale length 10 000 nominal, but the measurement includes a 6th digit (1nA) to preserve the best resolution in processing results.

Limits of Error: for 1 year, 20 ± 3°C.
At 10/s and 40/s: <± (0.02% rdg + 2 digits + 60nA)
At 100/s and 500/s: <± (0.02% rdg + 2 digits + 200nA)
Temp. coeff: < (0.0015% rdg + 1nA)/°C

AC VOLTAGE: true rms.
Nominal speed: 5/s
Speed selections (500, 100, 40 10/s) vary scale length only.
Scale length, for 500/s: 200
for 100/s: 2000
for 40 and 10/s: 20 000

Ranges:	20V§	2V	200mV	30mV
Nominal full scale:	20V	2.5V	312mV	39mV
Sensitivity, at 20 000 scale:	1mV	100μV	10μV	1μV

§Not available with solid-state selectors.

Limits of Error: 1 year, 20 ± 3°C
Crest factor: <4:1. Reading >10% of range.
40Hz to 10kHz: <± (0.5% rdg + 0.5% range)
20Hz to 20kHz: <± (2% rdg + 0.5% range)
Temp. coeff: <0.015% rdg/°C

AC CURRENT: true rms. Data for 100Ω shunt.
Using connector 35303B, which includes 100Ω shunts, quoted errors apply for currents <40mA. Connector 35303A may be used with external shunts.
Nominal speed: 5/s
Speed selections (500, 100, 40, 10/s) vary scale length only.
Scale length, for 500/s: 200
for 100/s: 2000
for 40 and 10/s: 20 000

Ranges:	200mA§	20mA	2mA	300μA
Nominal full scale:	200mA	25mA	3.12mA	390μA
Sensitivity, at 20 000 scale:	10μA	1μA	100nA	10nA

§ Not available with solid-state selectors.

Limits of Error: for 1 year, 20 ± 3°C.
Crest factor: <4:1. Reading >10% range.
40Hz to 10kHz: <± (0.5% rdg + 0.5% range)
20Hz to 20kHz: <± (2% rdg + 0.5% range)

RESISTANCE
Reading rates and scales as above

ENERGISATION 1mA, 4-terminal measurement.

Ranges:	10kΩ	1kΩ	100Ω	10Ω
Nominal full scale:	13.7kΩ	1.7kΩ	200Ω	25Ω
Sensitivity at 10 and 40/s:	100mΩ	10mΩ	1mΩ	1mΩ†

† scale length 10 000 nominal, but the measurement includes a 6th digit (100μΩ) to preserve the best resolution in processing results.

Limits of Error: for 1 year, 20 ± 3°C.
At 10/s and 40/s: <± (0.015% rdg + 2 digits + 6mΩ)
At 100/s and 500/s: <± (0.015% rdg + 2 digits + 20mΩ)

ENERGISATION 100μA, 4-terminal measurement

Ranges:	100kΩ	10kΩ	1kΩ	100Ω
Nominal full scale:	137kΩ	17kΩ	2kΩ	250Ω
Sensitivity at 10 and 40/s:	1Ω	100mΩ	10mΩ	10mΩ†

† scale length 10 000 nominal, but the measurement includes a 6th digit (1mΩ) to preserve the best resolution in processing results.

Limits of Error: for 1 year, 20 ± 3°C.
At 10/s and 40/s: <± (0.015% rdg + 2 digits + 60mΩ)
At 100/s and 500/s: <± (0.015% rdg + 2 digits + 200mΩ)
100kΩ range: <± (0.03% rdg + 2 digits)
Errors may exceed those quoted if measurements of high resistance are made in conditions of high humidity (e.g. >60% RH).

A-to-D CONVERTER

Input impedance: 100V dc and 20V ac: 1MΩ
other ranges: >1GΩ, <100pF
Input current: <200pA

INTERFERENCE REJECTION

Series mode, for 50(60)Hz ±2%: at 10/s: >70dB
at 40/s: >60dB

Effective common mode

With guard connected to source and 1kΩ imbalance,
for dc, at 10/s: >140dB
at 40/s: >140dB
for 50(60)Hz ±2%, reed selectors at 10/s: >140dB
at 40/s: >130dB
for 50(60)Hz ±2%, solid-state selectors
at 10 and 40/s: >120dB
Maximum common mode, reed selectors: 200V
solid-state selectors: 14V

PRT MEASUREMENT

20 ± 3°C, 40/s

4-terminal configuration, 1mA energisation, are arranged automatically. Linearisation to DIN 43760.*

Error in conformity, 1 year: −200 to 300°C: <0.2°C
300 to 600°C: <0.3°C
Repeatability, 24 hrs at 20°C:
−200 to 300°C: better than 0.03°C
300 to 600°C: better than 0.3°C

* This is expected to be published as new British and IEC standards.

THERMOCOUPLE MEASUREMENT

20 ± 3°C, 40/s

Cold junction compensation provided either by sensors within the input connector (ambient), or from junctions devised by the user. The cold junction temperatures and linearisation to IEC584 (BS4937) are processed automatically for the thermocouples listed below. Errors are over 1 year, to which must be added error in cold junction temperature.

Type	Composition	Range °C	Sensitivity °C	Error in conformity
E	NiCr/CuNi	−50 to 900	0.01	<0.4°C
J	Fe/CuNi	0 to 750	0.01	<0.35°C
K	NiCr/NiAl	−120 to 1370	0.01	<0.3°C
R	PtRh(13%)/Pt	0 to 1600	0.1	<1.0°C
S	PtRh(10%)/Pt	0 to 1600	0.1	<1.0°C
T	Cu/CuNi	−50 to 390	0.01	<0.35°C

Ambient junction error (0 to 50°C): <0.4°C

STRAIN MEASUREMENT

20 ± 3°C, 40/s

Full, ½- or ¼-bridge configurations occur automatically by front-panel selection. Dummies are 120Ω, or provided by the user.
Built-in remote dummy: (120Ω ± 0.01%) ± 5ppm/°C
Dual current system, energisation from 35301D. Calibration and calculation of με are automatic. Quoted performance is for 120Ω gauge, gauge factor 2, energisation 8.33mA, at 10 000με.
Sensitivity: 0.1με
Limit of error, 1 year: <15με
Repeatability, 24 hrs at 20°C: better than 2με

TRANSDUCER ENERGISATION MODULE 35301D

Voltage, for full-bridge strain: 2V
Setting error: <0.005%
Stability: drift over 1 year: <0.05%

Current:

Twin currents for resistance, PRT, ½- and ¼-bridge strain.
Values: 100μA, 1mA, 8.33mA
Setting error: <0.01%
Tracking error: <5 ppm/°C
Stability: drift over 1 year <0.005%
Leakage when switched off: <100pA
A fourth current, up to 10mA, may be defined by a resistor fitted by the user.

REED RELAY SELECTOR 35301A

Channels: 3-pole switching: 20
6-pole switching: 10
Any selector can handle a mix of 3-pole and 6-pole.
Isolation, channel to ground: <300pF >10GΩ
Channel input capacitance: <120pF
Crosstalk immunity: at dc: >120dB
at 10kHz: >80dB
Thermal error typically:* <2μV
Life, switching <10V <10mA: >10^8 operations
Maximum switched voltage: 200V peak
* Thermal Drift Compensation removes this error when making resistance-type measurements, e.g. PRTs and strain gauges.

SOLID-STATE SELECTOR 35301B

Channels: 3-pole switching: 20
6-pole switching: 10
Any selector can handle a mix of 3-pole and 6-pole.
Isolation, channel to ground: <1000pF >1GΩ
Channel input capacitance: <750pF
Crosstalk immunity: at dc: >120dB
at 10kHz: >80dB
Thermal error:* <2μV
Leakage current: <10nA
* Thermal Drift Compensation removes this error when making resistance-type measurements, e.g. PRTs and strain gauges.

Maximum ratings

Switched voltage: 14V peak
Overload protection: 30V
Transient protection: 100V

SLAVE EXPANSION UNIT 3530E

One, or two, slave units may be connected to ORION A or B. Each slave will house up to 10 reed relay selectors (35301A) or solid-state selectors (35301B). Performance when slaves are connected is the same as that given above, except:
Common mode rejection of dc and ac: subtract 10dB per slave
Channel crosstalk at dc: subtract 10dB per slave

Accuracy

AC voltage,
20V range, 10kHz to 20kHz: add 0.5% rdg per slave
Resistance, 100kΩ range: add 0.02% rdg per slave
These figures apply with Solartron cables not more than 1 m long.

EVENT-STATUS MODULE — 35302A

Channels: 20, opto-isolated
Isolation, each channel: 200V
Maximum input: 80V
Energisation: individually any input may be energised from the internal 5V supply, or from one of two external voltages.
External energisation supply: 4V to 12V or 12V to 80V

Events

Pulse input, minimum width: 2ms
Maximum rate processed:* 400 transitions/s
Events selectable: Lo to Hi, or Hi to Lo, or both
Event totalising: 65535 max
Event timing: ±2ms
*This is the maximum acquisition rate into memory. System throughput depends on tasks and processing that are programmed as well as the speed of interfaces and output devices.

Status

Level recognised: Logic 1 or Logic 0
Channel grouping: single or grouped bits
Module interfacing: 1-2-4-8 bcd
Reading rate: single bit 500 bit/s
as groups 1000 bit/s

DIGITAL INPUT-OUTPUT MODULE — 35302B

Input channels: 20 lines
Output channels: 20 lines
Inputs and outputs are floating with one common return.
Isolation: 100V

Inputs

Maximum input: +40V
Level recognised: CMOS compatible: Logic 0>3.5V, Logic 1<1.5V
Channel grouping: single or grouped bits
Module interfacing: binary 8-bit byte
1-2-4-8 bcd
Reading rate: single bit 500 bit/s
as groups 1000 bit/s

Outputs

All 20 outputs can operate in one of three ways:
1. Using internal + 5V supply (common 0V):
   Logic 0, sourcing 0mA: >4V
   Logic 1, sinking 16mA: <0.4V
2. Using external supply (common 0V):
   Maximum supply: +25V
   Logic 0, sourcing 0mA: supply voltage
   Logic 1, sinking 16mA: <0.4V
3. As On/Off switches (common 0V):
   Maximum rating, on: 150mA
   Maximum rating, off: +40V

Single channel: Hi or Lo alarm
Grouped channels: 8 max
Grouped channels can provide a sequence of 8 different output patterns, or can increment from one binary number to another.

COUNTER-TIMER MODULE — 35302C

Each channel has available both signal and gate inputs. Each input may be separately set to differential or TTL; either can be active Hi or active Lo.
The latest result is read when a channel is scanned.
Number of channels: 5
Differential sensitivity: 600mV
operating window: + 7V to – 7V about ground
Overload protection, signal: 25V max
common mode: 25V max

Limits of Error, for 1 year, 0° to 50° C: ± 0.005% rdg
Form of result: 5 significant digits
3 digits + exponent when $> 3 \times 10^7$

Frequency

Maximum input: 400kHz
Gate time, internal: 0.1, 1, 10 or 100s
external: 600s max
Speed of measurement update: 1/s max

Period

Maximum frequency: 400kHz
Number of periods measured: 1, 10, 100, 1000 or 10 000
Total time for periods selected: minimum 500μs
maximum 60 mins
Resolution: 1μs
Speed of measurement update: 1/s max

Interval

Time between positive and negative transitions, or the inverse.
Interval length: minimum 500 μs
maximum 60 mins
Resolution: 1μs
Speed of measurement update: 1/s max
Minimum time between intervals: 20ms

Totalise

Counts pulses in one of two ways:
1. A running total read on each scan.
2. A total accumulated during a gate input, reset to zero for each new gate.

Maximum input frequency: 400kHz
Minimum pulse width: 1.25μs
Maximum count: 32 bits ($>4 \times 10^9$)

ANALOGUE OUTPUT MODULE — 35301E

Channels: 5, isolated
Isolation, each channel: 200V

Voltage output: 0 to + 10V
Resolution: 12 bits (2.44mV)
Maximum current. 20mA
Current output: 4 to 20mA
Resolution: 4.88μA
Maximum voltage: 10V

Limits of Error, for 1 year at 20 ± 3°C: ±(0.1% rdg + 0.05% fs)
Temp. coefficient: < (0.01% rdg + 0.005% fs) /°C
Output noise: < 0.1% fs
Setting time to 1 bit: < 300ms

COMMUNICATION INTERFACES

The three types of interface enable ORION to be linked to a terminal at which full programming and control may be carried out, to a recording peripheral, or to a calculator or computer which may control and receive results.

Standard Serial Interface

Included in the Controller Module supplied with each ORION, the main intention of this interface is to provide full remote control from a terminal. It can also output results to a peripheral.

Electrical standard:	RS232C Type D
Connector:	25 way, Cannon type D
Data rate, switch selectable:	110 to 9.6 kbaud
Transmitted code:	ASCII 7-bit + even parity
Received code:	ASCII 7-bit, parity ignored
Maximum cable length:	15m (50ft)

SERIAL INTERFACE MODULE 35304B

This optional interface is primarily intended for high-speed communication with a computer.

Number of ports:	2
Electrical standard:*	RS422
Connector:	25 way, Cannon type D
Data rate, switch selectable:	300 to 155.6k baud
Transmitted code:	ASCII 7-bit + selectable parity
Received code:	ASCII 7-bit + selectable parity
Maximum cable length:	1.2km (4000ft)

*Each port may be independently set to RS232 Type D.

PARALLEL INTERFACE (GPIB) MODULE 35304A

This optional interface enables ORION to be connected on a general instrument bus where it may act as a talker or a listener on command. The switch is provided to set to talk only.

Electrical standard:	IEEE 488
Connector:	24 way, Amphenol
Parallel Poll sub-set:	PP1

PRINTER

Paper width:	62mm (2.44in)
Print speed:	2 lines/s
Characters per line:	20
Lines per channel, full format:	2
compact format:	1
Roll capacity:	approx 20 000 lines

TAPE RECORDER

Cartridge type:	DC100
Cartridge capacity:	500k byte
Channel records per cartridge, full format:	12 500
compact format:	25 000
encoded format:	50 000
Channel record speed, encoded format:	>100/s
Format:	as in ECMA46
Character blocks:	1024 byte
Replay time for full cartridge:	
via RS232 at 9.6k baud:	12 mins
via RS422 at 155k baud:	8 mins
Program storage: data per program:	16k byte
number of programs:	9

ENVIRONMENT

Temperature: operating:	0 to 50°C, 70% RH
storage:	−20 to 60°C
Vibration: operating:	0.5g, 20 to 200Hz
storage:	0.5g, 20 to 200Hz

Storage assumes cartridge not present.

POWER SUPPLY

AC mains, or external battery, or both may be connected.

Mains:	90 to 130V/180 to 260V, 47 to 63Hz, 70VA
Battery:	20 to 30V, 50W
Memory battery, NiCd rechargable:	life > 6 hrs

Power fail: external battery takes over if mains fails. Internal battery maintains memory and clock if external battery not present; automatic restart when power restored. If ORION is switched off, the internal battery maintains the clock for several months.

DIMENSIONS

Length:	610mm (24ins)
Width:	432mm (17ins)
Height (including 20mm feet):	241mm (9.5ins)
Weight, all modules fitted:	33kg (73lbs)

ACCESSORIES

The following are supplied with ORION:

Item	Qty	Part No
Mains lead	1	480140220
Fuses: 1A	2	360106320
2A	2	360106230
5A	2	360106250
Paper rolls	5	800100350
Tape cartridges	2	560000040
Rack mount brackets	2	35302038A
Module extractor	1	35300270A
Operating manual	1	35302040

The following are available as options:

Rack mount slide kit:	35305D
Technical manual	35302036

Error No.	Error	Remedy
174	Run Time: channel not present	Either correct the Channel number in the program, or add the Channel to the Task Definition.
175	Run Time: formula too complex, i.e. more than 4 pending operations	Split the formula into two two separate lines of program.
199	As for 171 and 172	Logger requires servicing.
253	No energisation module	A Transducer Energisation Module (35301D) must be fitted in order to measure the sensor type specified.
254	Channel number conflict	The Channel numbers specified do not correspond (wholly or partially) with the type or quantity of Input Selectors fitted. Enter the correct Channel numbers.
255	Function not applicable (e.g. Alarms not applicable to event Channels)	The requested function is not available.

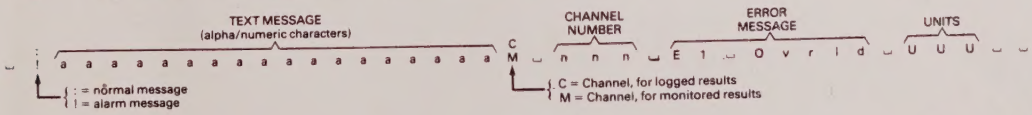
2 Measurement errors included in the output results

If an error is detected, one of the following 9 error messages will be output instead of the numerical value of the result.

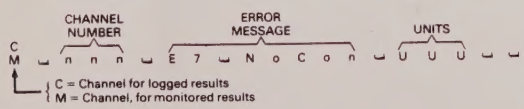
ERROR MESSAGE	ERROR INDICATED
E1 Ovrld	Measurement overload
E2 Ovrld	Measurement out of linearisation range
E3 Ambnt	Invalid ambient temperature
E4 Ovrfl	Event total exceeds 65536
E5 TmOut	Timeout in external module measurement (i.e. > 20 s)
E6 Initl	Strain gauge channel not initialised
E7 NoCon	Input connector not fitted
E8 Wait	Result not yet accumulated (Counter Timer Module)
E9 Off	Counter Timer Channel is off

Error messages are included in Full Format and Compact Format results, see the examples below.

e.g. 1. Typical error message in a Full Format result.



e.g. 2 Typical error message in a Compact Format result.



The 9 error messages E1-E9 correspond with the 9 error codes in the Tag Bytes of Coded Format results, see page 11.19.

3 TAPE HANDLING MESSAGE

When replaying data from the ORION cartridge one of the following messages may be output to the controlling device.

Messages	Reason
*old format	Tape not recorded on 3530D ORION
*tape fault	Cartridge not in reader
*write protected	Cartridge write protected
*end of tape	End of tape reached
*end of data	End of data reached
*short block	Recorded data corrupted or faulty tape
*CRC error	
*bad format	

Note: The first character of the error message will be a space character, the second character will be an asterisk.

Solartron Instruments

Solartron Instruments
Victoria Road, Farnborough, Hampshire GU
Telephone: Farnborough (0252) 544433
Telex: 858245 Solfar G.
Cables: Solartron Farnborough
A division of Schlumberger Electronics (UK

MP MASTER
MA20216
SOLARTRON
Schlumberger